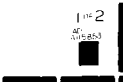
AD-A115 853 MITRE CORP BEDFORD MA F/G 9/2
USER-SYSTEM INTERFACE DESIGN FOR COMPUTER-BASED INFORMATION SYS--ETC(U)
APR 82 S L SMITH F19628-81-C-0001
UNCLASSIFIED MTR-8464 ESD-TR-82-132 NL



ESD-TR-82-132

MTR-8464

USER-SYSTEM INTERFACE DESIGN

FOR

COMPUTER-BASED INFORMATION SYSTEMS

By

SIDNEY L. SMITH

APRIL 1982

Prepared for

DEPUTY FOR TECHNICAL OPERATIONS AND PRODUCT ASSURANCE
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
Hanscom Air Force Base, Massachusetts

Project No. 572C
Prepared by

THE MITRE CORPORATION
Bedford, Massachusetts

Contract No. F19628-81-C-0001

Akiko P. Sudano

AKIKO P. SUDANO, 2Lt, USAF
Project Officer

James W. Neely, Jr.

JAMES W. NEELY, JR., Lt Col, USAF
Chief, Computer Engineering
Applications Division

FOR THE COMMANDER

Walter W. Turgiss

WALTER W. TURGISS
Acting Director, Engineering and Test
Deputy for Technical Operations
and Product Assurance

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESD-TR-82-132	2. GOVT ACCESSION NO. A115853	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* USER-SYSTEM INTERFACE DESIGN FOR COMPUTER-BASED INFORMATION SYSTEMS		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER MTR-8464
7. AUTHOR(s) SIDNEY L. SMITH		8. CONTRACT OR GRANT NUMBER(s) F19628-81-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS The MITRE Corporation P. O. Box 208, Bedford, MA 01730		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Project No. 572C
11. CONTROLLING OFFICE NAME AND ADDRESS Deputy for Technical Operations and Product Assurance Electronic Systems Division, AFSC Hanscom Air Force Base, MA 01731		12. REPORT DATE APRIL 1982 13. NUMBER OF PAGES 190
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

COMPUTER-BASED INFORMATION SYSTEMS
DESIGN GUIDELINES
REQUIREMENTS DEFINITION
USER-SYSTEM INTERFACE

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

For a broad range of computer-based information systems, the user-system interface (USI) can require a sizable investment in software design. In current practice, there is no coherent methodology for USI design, but efforts to establish USI design standards have begun. This report summarizes the present state of the art, including a recent survey of USI design practice (Smith, 1981b), and proposes several aids to USI design.

The first step in USI software design is to decide what is needed. A structured checklist of USI functional capabilities is proposed (over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. (concluded)

to aid in requirements definition. This checklist updates a previously published version (Smith, 1981a).

Once USI requirements have been defined, the next step is to provide design guidance for the software needed to implement those requirements. A compilation of 375 design guidelines is offered here for major USI functional areas of data entry, data display, and sequence control, revising and enlarging previously proposed guidelines (Smith, 1981a).

A third step leading to good USI design is careful documentation. USI documentation is needed to permit design review before software implementation, and continuing design coordination thereafter. Potential computer aids are briefly described for generation of draft USI functional specifications, the earliest stage of design documentation.

ACKNOWLEDGMENTS

This report has been prepared by The MITRE Corporation under Project No. 572C. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Massachusetts.

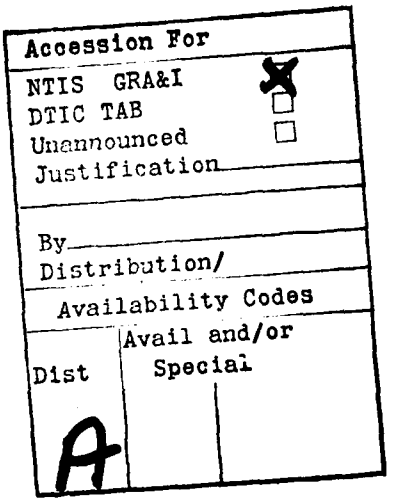


TABLE OF CONTENTS

		Page
LIST OF ILLUSTRATIONS		vi
SECTION 1	INTRODUCTION	1
SECTION 2	BACKGROUND	3
	INFORMATION SYSTEMS	3
	USER-SYSTEM INTERFACE (USI)	4
	USI SOFTWARE	6
	DESIGN DEFICIENCIES	7
	DESIGN PRACTICE	8
SECTION 3	USI REQUIREMENTS DEFINITION	11
	GENERAL	11
	WORK ENVIRONMENT	11
	INTERFACE HARDWARE	12
	USER CHARACTERISTICS	13
	TASK ANALYSIS	17
	FUNCTIONAL CAPABILITIES CHECKLIST	20
	USE OF THE CHECKLIST	22
SECTION 4	USI DESIGN GUIDELINES	24
	CURRENT STATUS	24
	GUIDELINES COMPILATION	27
	FUTURE DEVELOPMENT	29
SECTION 5	USI DOCUMENTATION	32
	CURRENT STATUS	32
	SPECIFICATION GENERATION	33
	DESIGN DOCUMENTATION	36
	DESIGN REVIEW	38
	DESIGN COORDINATION	39
REFERENCES		42
APPENDIX A	USI FUNCTIONAL CAPABILITIES CHECKLIST	47
APPENDIX B	DESIGN GUIDELINES FOR DATA ENTRY FUNCTIONS	67
APPENDIX C	DESIGN GUIDELINES FOR DATA DISPLAY FUNCTIONS	91
APPENDIX D	DESIGN GUIDELINES FOR SEQUENCE CONTROL FUNCTIONS	129
APPENDIX E	REFERENCES FOR DESIGN GUIDELINES	175

LIST OF ILLUSTRATIONS

		Page
Figure 1.	Factors Influencing User-System Interface Design	5
Figure 2.	Steps in the USI Design Process	18
Figure 3.	Future USI Guidelines Development	30
Figure 4.	Sample Output of Patterned Prose	35

SECTION 1

INTRODUCTION

In system acquisition programs managed by the Air Force Electronic Systems Division (ESD), special attention must be given to design of the user-system interface (USI). In recent years, two ESD reports have dealt with this subject, recommending improvements to USI design methodology (Smith, 1980a; 1981a). Work in this area is continuing under MITRE Project 5720 for USI requirements definition, and under Air Force Program Element PE 64740F (MITRE Project 5220) for the development of USI design guidelines.

This present report supersedes the two previous reports noted above. The earlier text has been enlarged and updated to include results from a recent survey of USI design issues (Smith, 1981b). Other material from the earlier reports has been preserved and restated here to provide comprehensive coverage of the subject.

Discussion of information systems is couched in general terms throughout this report, to acknowledge that USI design problems and their possible solutions are common to many different systems. Improvements to USI design methodology proposed here will be evaluated in their specific application to ESD/MITRE development of Air Force command, control and communication systems.

Section 2 of this report provides background discussion of the user interface in information systems, and particularly the important role of USI software. It is argued that poor USI design can contribute to system failure, and that poor design may result from deficiencies in current USI design practice.

Section 3 discusses the process of defining USI requirements. Particular emphasis is given to the need for task analysis to determine data handling requirements with implications for USI software design, and the interpretation of those requirements in terms of functional capabilities required for the user interface. A checklist of USI functional capabilities, hierarchically organized, is proposed as an aid for defining USI requirements.

The current version of that checklist is presented in Appendix A to this report. The checklist has been modified in minor respects from previously published versions.

Section 4 discusses means for providing more effective guidance to USI design, with emphasis on the design of USI software. Available USI design guidelines, in scattered published sources, provide only partial coverage of limited usefulness. It is recommended that a comprehensive compilation of USI design guidelines be undertaken as a collaborative effort by people concerned with this problem.

Examples of guidelines that should prove useful in USI design are provided as appendices to this report. Appendix B lists design guidelines pertaining to data entry functions. Appendix C lists guidelines for data display, an initial compilation for this report. Appendix D lists guidelines for sequence control. The two previously published lists, those for data entry and sequence control, have been revised and expanded somewhat in this new version, following suggestions made by reviewers.

Section 5 discusses USI design documentation. Improvements in USI documentation are needed, but no specific recommendations are made at this time. In a current experiment, the USI functional capabilities checklist is being redesigned as an on-line computer aid for automatic conversion of checklist entries into draft written specifications, a technique which may offer some promise for the future.

This report concludes with a list of referenced documents. A separate listing of reference sources cited in connection with design guidelines is provided in Appendix E.

Work is continuing at ESD/MITRE on the development, application and evaluation of tools for USI requirements definition (the checklist) and for design guidance (the guidelines). This work would benefit from the help of people in other organizations as well. Can you propose additions or changes to the checklist, or to the design guidelines? Can you apply and help evaluate these design tools in your own work? If so, please contact:

Sidney L. Smith
The MITRE Corporation
Bedford, Massachusetts 01730

SECTION 2

BACKGROUND

For a broad range of computer-based information systems, the user-system interface (USI) can require a sizable software investment and may suffer from design deficiencies that handicap system performance. No comprehensive guidance is currently available in design practice, but efforts to develop USI design standards have begun.

INFORMATION SYSTEMS

Current use of computers covers a broad range of applications. At one extreme are large, general-purpose computer installations, used by different people for different purposes. Users are expected to provide exact instructions (a computer program) in order to request data processing. Many users have programming skills. All users must have some knowledge of the system and its capabilities.

At the other extreme are the small, special-purpose computers used increasingly as components within larger systems. Such component computers may regulate the ignition of an automobile, or the tuning of a television set. Their net effect is to help make system operation easier. The user does not interact directly with component computers, and indeed may not even know they are there.

Between these extremes of general and very specialized application, computers are used to support a wide variety of what are commonly called information systems. Information systems are task-oriented rather than general purpose, being designed and dedicated to help perform defined jobs. Applications range from relatively simple data entry and retrieval (e.g., airline reservations) through more complex monitoring and control tasks (inventory control, process control, air traffic control), to jobs requiring long-term analysis and planning. Military command systems fall within this range of applications.

Many information systems are event-driven, with consequent time pressure on their users. For such systems, job performance can be facilitated by good design, or handicapped by poor design. Because system design will have a critical effect on performance, required data handling capabilities are usually defined by functional specifications established in advance of design.

The users of information systems must interact with a computer in some explicit fashion to accomplish the data handling tasks needed to get their jobs done. The users of information systems differ in ability, training and job experience. Users may be keenly concerned with task performance, but they probably have little knowledge of (or interest in) their computer tools. Design of the user-system interface must take these human factors into account.

USER-SYSTEM INTERFACE (USI)

What is the user-system interface? Here the phrase is defined broadly to include all aspects of system design that affect a user's participation in data handling transactions. A variety of factors are involved in the user-system interface, as indicated in Figure 1.

Directly observable aspects of the USI include the physical work environment and the hardware facilities at the user's work station. Such physical aspects, sometimes called the man-machine interface, have been the subject of conventional human engineering study, where the concern is for proper illumination, seating, workplace arrangement, keyboard layout, display contrast, symbol size, etc. Good design of the physical workplace is important, of course, but by itself is not sufficient to ensure effective job performance.

Also important are less tangible aspects of information system design -- the ways in which data are stored and manipulated, including paper files and forms, if any, and the procedures and processing logic that govern data handling transactions. Forms, procedures and logic involve software design, the design of computer programs to permit hardware (and paper) to be used in conjunction with automated data processing.

If the USI is conceived in these broad terms, to encompass all factors influencing user-system interaction, then to say that the USI is critical to successful system operation is to state the obvious. In any automated information system, whether its work stations are used for data input, calculation, planning, management or control, effective USI design is required for effective performance. Task analysis, review of operating procedures, equipment selection, workspace configuration, and especially USI software design -- all must be handled with care.

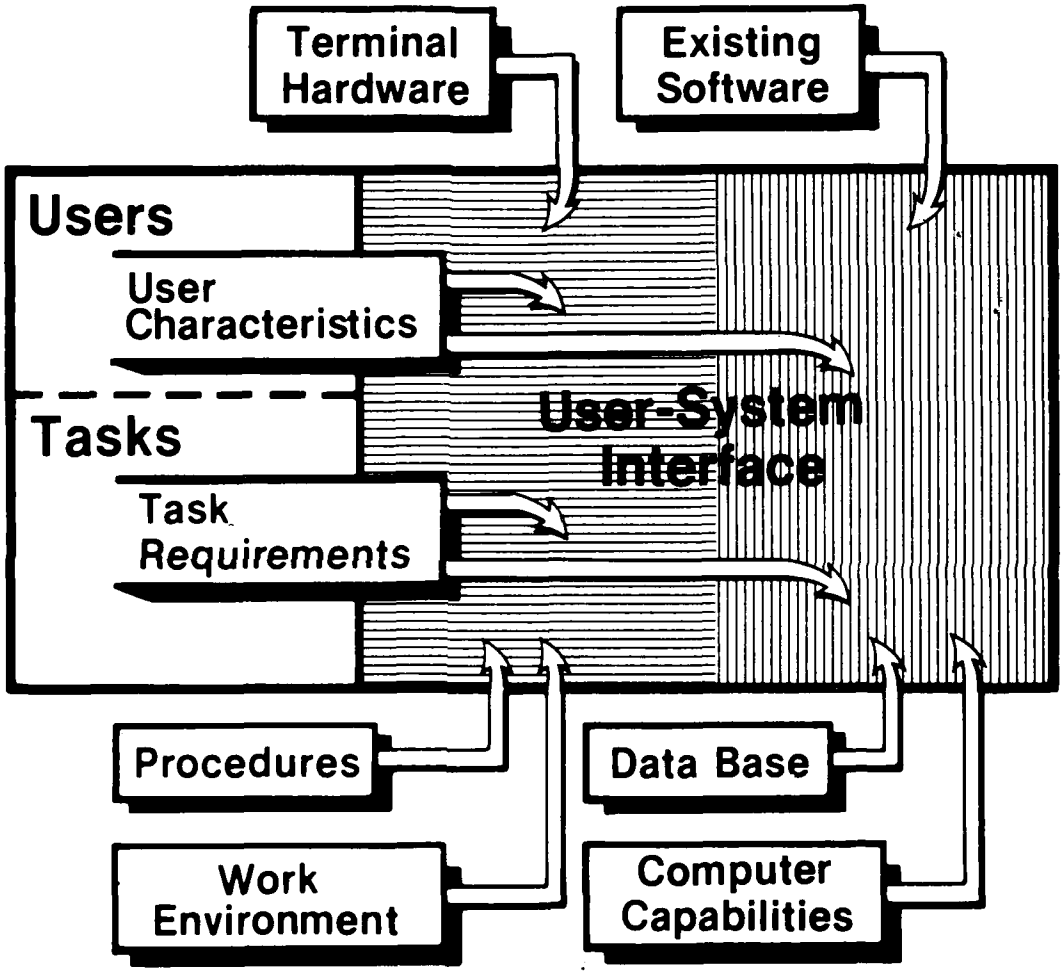


Figure 1. Factors Influencing User-System Interface Design

USI SOFTWARE

The critical role of USI software in system design poses a special challenge to human factors specialists, recognized a decade ago by Parsons:

> . . . what sets data processing systems apart as a special breed? The function of each switch button, the functional arrangement among the buttons, the size and distribution of elements within a display are established not in the design of the equipment but in how the computer is programmed. Of even more consequence, the 'design' in the programs establishes the contents of processed data available to the operator and the visual relationships among the data. In combination with or in place of hardware, it can also establish the sequence of actions which the operator must use and the feedback to the operator concerning those actions.
>
> (1970, page 169)

Current interest in USI software design is reflected in phrases such as "software psychology" (cf. Shneiderman, 1980). But USI design cannot be the concern only of the psychologist or the human factors engineer. It is a significant part of information system design that must engage the attention of system developers, designers, and ultimately system users as well.

Not only is USI software design critical to system performance, it can also represent a sizable investment of programming effort during initial system development, and during subsequent system modification ("software maintenance") to accommodate changing operational requirements thereafter. Just how sizable is that investment? The answer, of course, will depend upon the particular type of information system being considered.

In a recent survey (Smith, 1981b), people involved in information system design were asked to estimate the percent of operational software devoted to implementing the USI. Respondents were asked to give minimum, average and maximum estimates for the type of system they were describing. Mean estimates for 31 respondents were 18.3 percent "as a minimum", 34.3 percent "on the average", and 47.7 percent "at the most". That is to say, the respondents postulated a sizable variability in their on-the-average estimates. Overall, the average estimate that USI design comprises a bit more than 30 percent of operational software seems a reasonable figure.

As it happened, each respondent referred to a different type of system, and their estimates differed accordingly. The respondent offering the lowest estimates gave an on-the-average figure of 3 percent. At the other extreme, the respondent offering the highest estimates gave and on-the-average figure of 80 percent. This range reflects common experience that some computer systems require a much higher investment in USI design than others, depending upon their purpose.

DESIGN DEFICIENCIES

In every information system there are probably deficiencies in USI design, some major, some minor. This is not the place to belabor the point with a compendium of bad examples. The topic has been dealt with by Ramsey and Atwood (1979), and by other writers. But a general discussion of the potential consequences of poor USI design is probably pertinent here.

Given the broad definition for the user-system interface that was proposed above, it is obvious that deficiencies in USI design may result in degraded system performance. To be sure, users can sometimes compensate for poor design with extra effort. Probably no single USI design flaw, in itself, will cause system failure. But there is a limit to how well users can adapt to a poorly designed interface. As one deficiency is added to another, the cumulative negative effects may eventually result in system failure, poor performance, user complaints and/or developer dissatisfaction.

Outright system failure may sometimes be seen in abandoned installations, or decreasing use of an automated information system where use is optional. There may be retention of (or reversion to) manual data handling procedures, with little use of automated capabilities. If a system does fail in this way, the result is disrupted operation, wasted time, effort and money, and failure to achieve the potential benefits of automated data handling.

In a constrained environment, such as that of many military and commercial information systems, the user may have little choice but to make do with whatever interface design is provided. Here the symptoms of poor USI design may appear in degraded system performance. Frequent and/or serious errors in data handling may result from confusing USI design. Tedious user procedures may slow data processing, resulting in longer queues at the checkout counter, the teller's window, the visa office, the truck dock, or any other workplace where the potential benefits of automation are outweighed by an unintended increase in human effort.

In situations where degradation in system performance is not so easily observed, symptoms of poor USI design may appear as user complaints. The system may be described as hard to learn, or clumsy, tiring and slow to use. The user's view of the system is conditioned chiefly by experience with its interface. If USI design is unsatisfactory, the user's image of the system will be negative, regardless of any niceties of internal computer processing.

USI design may prove deficient from the broader perspective of system developers, who are sometimes disappointed by discrepancies between what was intended and what is actually achieved. Different portions of the USI design may be implemented by different people, who share no common view of desired operational procedures. The probable result is design inconsistency, forcing users to handle different tasks differently, or even different transactions within the same task, and thus imposing an unnecessary burden on both learning and use of the system. System developers will be justifiably critical of such design inconsistency, on behalf of the eventual system users.

System developers may also worry about design inconsistency among different systems. Within a large industrial firm, or within a government agency or a military service, many different information systems may be developed for different purposes in different parts of the organization. Where USI design is established independently for each system, the result could be to impose a burden of incompatible habits on the user who must move from one system to another. Conversely, some standardization of USI design across systems might help to reduce user learning problems and improve user performance.

Examples of inter-system inconsistencies of USI design are currently being documented in a study of battlefield information systems, sponsored by the Army Research Institute (ARI, 1979; Sidorsky and Parrish, 1980). The objective of that study is to develop USI design guidelines to help ensure system inter-operability, from the user viewpoint, as well as to improve the design of individual systems.

DESIGN PRACTICE

It seems fair to characterize present methods of USI software design as art rather than science, depending more upon individual judgment than systematic application of knowledge (Ramsey and Atwood, 1979; 1980). As an art, USI design is best practiced by experts, by specialists experienced in the human engineering of computer systems. But such experts are not always available to help

guide system acquisition, and certainly cannot guide every step of USI design at first hand. What seems needed is some way to embody expert judgment in the form of explicit procedures and guidelines for USI design.

USI design guidance is needed at several stages of system development, to define USI software requirements in system functional specification, to provide guidelines before USI software design, and to permit design verification before implementation of USI software. Unfortunately, there is little effective guidance for USI design in present practice.

Consider the development of military information systems. Military Specification MIL-H-48655B (1979) calls for a system development sequence starting with requirements analysis, functional specification and design verification. Actual USI software design in system development will sometimes depart from this desired sequence. There may be no explicit attempt to determine USI requirements. Specifications may include only rudimentary references to USI design, with general statements that the system must be "easy to use". In the absence of more effective guidance, both the design and implementation of USI software may become the responsibility of programmers unfamiliar with operational requirements. USI software may be produced slowly, while detection and correction of design flaws occur only after system prototyping, when software changes are difficult to make.

Human engineering standards and design handbooks have been of little use to the software designer. MIL-STD-1472B (1974), a major human engineering design standard for system procurement, was concerned almost exclusively with hardware design ("knobs and dials") and physical safety ("avoid sharp corners"). The recently published human factors design handbook by Woodson (1981) contains only three pages of general material on information processing (in a total of approximately 1000 pages) and contains no reference to computer systems in its index.

Some guides to USI software design might be reasoned by analogy from hardware standards. Just as we should not design sharp corners on hardware, we should not include hazardous features in user software. Just as a physical step may be marked with a painted line to keep us from tripping on it, so we may seek some way to signal abrupt shifts in the logical steps of an interactive sequence. A pushbutton reserved only for emergencies may be physically shielded to prevent accidental activation. So too it is possible to provide software protection in the USI to prevent accidental initiation of critical actions. But such analogies do not take us very far.

MIL-STD-483 refers only briefly to USI software design, indicating that human performance/human engineering requirements should specify "minimum times for human decision making, maximum time for program responses, maximum display densities of information, clarity requirements for displays, etc." (1970, page 43, paragraph 60.4.3.2.2.1)

MIL-STD-454F offers just one paragraph on the general subject of human engineering:

> Human engineering design criteria and principles shall be applied in the design of electronic equipment so as to achieve safe, reliable, and effective performance by operator, maintenance and control personnel, and to minimize personnel skill requirements and training time. MIL-H-46855 shall be utilized as a guideline in program planning and MIL-STD-1472 as a guideline in applying human engineering design criteria. Quantitative human engineering requirements shall be as specified in the contract.
>
> (1978, page 62-1)

In 1981, MIL-STD-1472 was published in a revised "C" version. That new military standard includes nine pages dealing with USI software design, in Section 5.15 titled "Personnel-Computer Interface". Thus a modest beginning has been made, but much more is needed. There are still no comprehensive guidelines available. The question is, can needed USI design guidance be developed?

Air Force efforts to provide USI design guidance (Smith, 1980a; 1981a) have emphasized three aspects. First, USI functional requirements should be defined early in system development. Second, design guidelines should be established in relation to required functional capabilities. Third, USI design should be documented in a form to permit design review and verification prior to software implementation. These three aspects of USI design guidance are discussed in succeeding sections of this report.

SECTION 3

USI REQUIREMENTS DEFINITION

The first step in USI software design is to decide what is needed. USI requirements definition must consider the work environment, the interface hardware, the characteristics of the people who will use the system, the data handling requirements of their jobs, and the functional capabilities of the USI that are needed in order to perform those jobs. A structured checklist of USI functional capabilities is proposed to aid requirements definition.

GENERAL

There is a generally held belief among people involved in system development: the better you can define what is required, the better the resulting design will be. Conversely, poor specification of system requirements may lead to poor design.

In a recent survey (Smith, 1981b), 59 percent of people concerned with USI design judge that USI requirements are not adequately defined in system specifications. Only 15 percent report adequate coverage. Many respondents, 68 percent, cite deficiencies in system specification and design related to inadequate early attention to USI requirements.

How can USI requirements definition be improved? A variety of factors influencing USI design must be considered. Three frequently cited factors are work environment, interface hardware, and user characteristics. These factors are discussed separately in the paragraphs below.

It can be argued, however, that all of these factors are subsumed in some degree in task analysis of the data handling functions that must be performed in an information system. The balance of this section will discuss task analysis, and the possibility of developing a checklist of functional capabilities to help move from task analysis to USI requirements definition.

WORK ENVIRONMENT

In itself, the design of a work environment seldom fulfills functional requirements directly. Instead, the general objective of workspace design is to minimize constraints on functional

performance. In many information systems, the immediate work environment is relatively benign, and does not constrain USI design. But there are exceptions. For example, the noise, dirt and confusion of movement at a truck dock may limit USI options available for air cargo data entry (Smith 1975; 1976).

Even in clean, quiet office locations, poor workplace layout can handicap job performance. Inaccessibility of paper files, inadequate space at the user's work station, badly positioned displays and keyboards, glare sources in unbalanced ambient illumination, all take their toll. It can be argued that environmental deficiencies such as these contribute in large measure to user complaints of fatigue after long hours at a constrained work station.

When an optimal work environment is not possible, some limits on USI design may have to be imposed in order to accommodate sub-optimal working conditions. As an example, mechanical vibration in airborne command posts might hinder fine manipulations required for lightpen use. A noisy environment might obscure voice output displays and other auditory signals. A high ambient illumination, such as experienced in an open control tower on a sunny day, might reduce the effectiveness of visual displays, unless appropriate shielding is provided.

Where a normal working environment can be assumed, it may be possible to rely on standard procurement specifications and engineering practice to produce a good design. Where unusual working conditions must be accommodated, however, it will be necessary to include a description of environmental constraints in the USI design specification.

INTERFACE HARDWARE

Although it is usually the logic and software implementation of USI design which prove critical, hardware choices can affect design implementation and eventual system performance. Hardware here is meant to include input devices, output display and signaling devices, and also printouts, paper forms and other equipment which may be used in conjunction with the USI. If the USI is used to mediate user-to-user data transfer and message exchange, which is becoming more common, then communication facilities should probably also be included under this heading.

Functional specifications for hardware, such as those for display capacity, legibility, etc., are reasonably well understood in system development and will not be considered further in this

report. Such device capabilities may be included in a USI functional specification, however, if physical requirements can be determined in advance of USI design. Ideally, hardware specification should follow and derive from functional design. In practice, it may happen that system development must build upon and hence be constrained by existing equipment.

The effect of hardware and associated procedural constraints on USI design may be more subtle than suggested by physical equipment specifications. The technique and format of data input may have to accommodate the nature of the data source, paper forms being transcribed, etc. For data output, display formats may have to be adaptable to document printing and other constraints. For sequence control, hardware choices can have a pervasive influence. As an obvious example, the availability of a lightpen or other pointing device for selection of control options will permit more flexible display formatting than the use of multi-function keys which are labeled in display margins.

Where anticipated hardware limitations will constrain USI functional capabilities, such constraints should be indicated clearly in the design specification, so that functional design can be compromised as necessary. Where USI hardware can be chosen freely, as in acquisition of a new system, it may be desirable not to impose any special constraints except those implied in equipment design standards. The inclusion of detailed equipment "requirements" in system specification may sometimes be supposed an adequate substitute for the lack of detailed functional specifications. In USI design, however, with its heavy dependence on software for effective implementation, detailed hardware specification may make good design harder to achieve rather than easier.

USER CHARACTERISTICS

A challenging aspect of information system design is the broad range of user characteristics that must be accommodated, including people with widely different skills, training and experience -- managers as well as clerical personnel, hardware and software technicians performing maintenance functions, even "box kickers" at a truck dock in the automation of air cargo data handling (Smith, 1976).

There are various ways to categorize the different kinds of users. One categorization distinguishes "operators" and "analysts" from "service" personnel (Goodwin, 1978; see also Rouse, 1975). In this view, an operator is a person performing a structured task,

usually at a forced pace, monitoring and controlling through system outputs and inputs, making decisions and initiating actions through the system within limited time constraints: e.g., radar track monitoring; air traffic control; process control. An analyst performs an unstructured task, usually self-paced, manipulates system data to establish relationships, perhaps using on-line tools, may rely also on data from other sources including past experience to recognize problems and make decisions, which may not be made immediately and may not be entered into the data processing system for action: e.g., intelligence analysis; mission planning; message handling. Service personnel perform more routine clerical tasks, where the user is not the source of data being entered, data retrieval is in response to specific query, transaction sequences are highly structured, and may be performed at a forced pace under pressure: e.g., making airline reservations; providing telephone directory assistance; word processing jobs.

Although such categorization offers helpful descriptive information of the user's role, it is largely differences among jobs rather than differences among people that are being characterized. Job differences can be defined better in the more specific context of task analysis. In this present report, there will be no attempt to label user differences in terms of job descriptions. No distinction is intended here between the terms "user" and "operator".

An important distinction can be made between dedicated and casual users (Martin, 1973). Information systems include both kinds. A "dedicated" operator in a surveillance job may spend virtually full time monitoring computer-generated displays. Such a person will generally receive extensive training, and the USI design can be tailored to his presumed skills. A "casual" user might be an intelligence analyst who interacts with a computer system only periodically, and whose requirements for information retrieval are wide ranging. Although this person may have been instructed in USI procedures, perhaps being given a user's manual, his use of computer aids may remain uncertain for an extended period unless on-line guidance is incorporated in USI design. Although these examples are cited to illustrate user characteristics, note again that they actually reflect important differences in job requirements.

For any one category of user, individual differences in skill may be considerable. In military systems, because of systematic rotation of job assignments for personnel, today's naive user becomes next year's expert, then to be replaced by another beginner. This is true of many non-military systems as well. Such regular personnel turnover implies the need for flexible job aids in USI design, which can provide optional help to the novice user but which can be bypassed by the expert.

Where there is high personnel turnover, low skill levels and minimal user training, USI design must take that into account. Interactive sequences should be configured as a routine series of simple steps, with adequate guidance and on-line error correction procedures to ensure effective user performance. For most applications, however, both user characteristics and job demands will require USI design offering more extensive capabilities, and permitting more flexible use.

As a general objective, it can be argued that the USI design in automated information systems should not require higher skills and greater user effort than the manual procedures which are being replaced. In practice this design objective is not always attained. Some automated systems prove harder to use rather than easier, and make increased demands upon their users.

In the Army RFP cited earlier, the increasing demand for skilled users caused by command automation is advanced as an argument in favor of developing USI design guidelines:

> The skill/demand mismatch is due in part to the fact that the equipment/procedural configurations of existing and projected systems have been devised without coordination among proponents of different systems. As a result, very little of the skill and know-how accrued from experience with one system can be transferred to other systems. Therefore, one of the long range goals of this project is to promote <u>functional</u> standardization and modularization of user/operator tasks and procedures in order to reduce the amount of training and the skill levels required of users/operators of fielded automated data processing systems. Many of the procedures employed in the operation of various automated systems are basically similar. Yet from the user/operator's perspective each system is a new situation with little carryover or transfer from previous exposures to other systems. While it may be too early to establish absolute "standards" for many system parameters, a measure of consistency would go a long way toward increasing effectiveness, reducing personnel costs and making battlefield automated systems more approachable to Army users/operators.
>
> (ARI, 1979)

This call for consistency is common to all recommendations on USI design, but is particularly important in settings that involve frequent rotation of personnel. To the extent that jobs may be similar in different information system applications, the operator transferred from one system to another ought to be able to use

similar means to accomplish similar ends. Even for quite different system applications, it may prove possible to introduce a coherent general approach to USI design, so that the interface logic will seem familiar to a person transferring to a new job.

If USI design is tailored to a particular user, as a personal interface, then that individual's characteristics and preferences can be taken into account. This is not usually done in information system design. The future users are often unknown to the designer, and in any case will probably be a mixed group. Under these circumstances, accommodation of individual user differences can only be provided in terms of flexible interface capabilities, permitting each user some adjustment of USI design to individual needs.

If a system is to be used by a group with special characteristics -- blind people, perhaps -- then USI design should be modified accordingly. An example of this would be adaptation of command languages for users with different linguistic backgrounds.

By and large, however, general user characteristics are assumed in system design, within the limits of variability implied by each job. To know the job is to know a great deal about the people who will perform that job. So the emphasis in USI design soon shifts from consideration of user characteristics to analysis of job requirements, which is discussed next.

Before leaving this subject, however, it may be helpful for the developers and designers of information systems, to postulate some general characteristics of prospective users. Assume that users will be intelligent men and women with their own special skills. These people will not necessarily be knowledgeable about computer technology, may have little time to learn complicated interface procedures, and will have different degrees of familiarity with the system. Being human, these people will sometimes make mistakes, especially when working under pressure, and good USI design must take that into account.

These people are usually motivated toward effective job performance in the face of operational demands. They will regard automated data processing as a tool to aid job performance, with little curiosity about the internal mechanisms of computing machines. Users will tend to judge the entire system on the basis of their personal experience with its interface. If the USI is efficient and easy to use, they will like the system. But users will be impatient and critical when handicapped by a clumsy interface design.

TASK ANALYSIS

Fundamental to USI design is the analysis of user tasks. Some general discussion is required here, to clarify both concepts and terminology. A simplified summary of the design process is presented in Figure 2. As indicated in that figure, analysis begins with definition of the mission requirements of a proposed system, the basic objectives to be accomplished. Those mission requirements are then elaborated and translated, taking into account the proposed operational employment concept, environmental, technological and fiscal constraints, to define system operational requirements.

Operational requirements imply the performance of various identifiable functions -- data sensing, data transmission, data processing, etc. Analysis of those functions, in turn, establishes more specific data processing requirements -- what data must enter the system, what data must be stored, what combinations and transformations of data are required, what kinds of information should result from that processing.

Those data processing functions imply the specification of tasks to accomplish particular ends. Some tasks may be performed entirely by machine and thus affect USI design only indirectly if at all. Because of the critical role of human judgment, however, and the fact that much of the data to be processed must be generated and/or evaluated by people, many tasks will involve the participation of system users.

Tasks are often labeled in operational terms, i.e., as activities a user must do -- scheduling, monitoring, text editing, etc. In the course of task analysis, most data handling tasks can be partitioned further into identifiable subtasks. Subtasks of text editing, for example, would include such activities as adding or deleting text, copying or moving text, paragraphing and pagination.

Subtasks, in turn, are often designed as a sequence of simple, discrete transactions, such as entry of a single item of data. As defined here, note that a transaction is the smallest functional "molecule" of user-system interaction. Other writers may adopt different terminology (cf. Pew and Rollins, 1975), but in this report "transaction" refers to an input by the user followed by a response from the computer.

The analysis of tasks into discrete transactions can be done with detailed operational sequence diagrams, or other similar methods. Here the designer tries to predict user actions under different circumstances, and gain insight into the data handling transactions required for successful task performance. Each

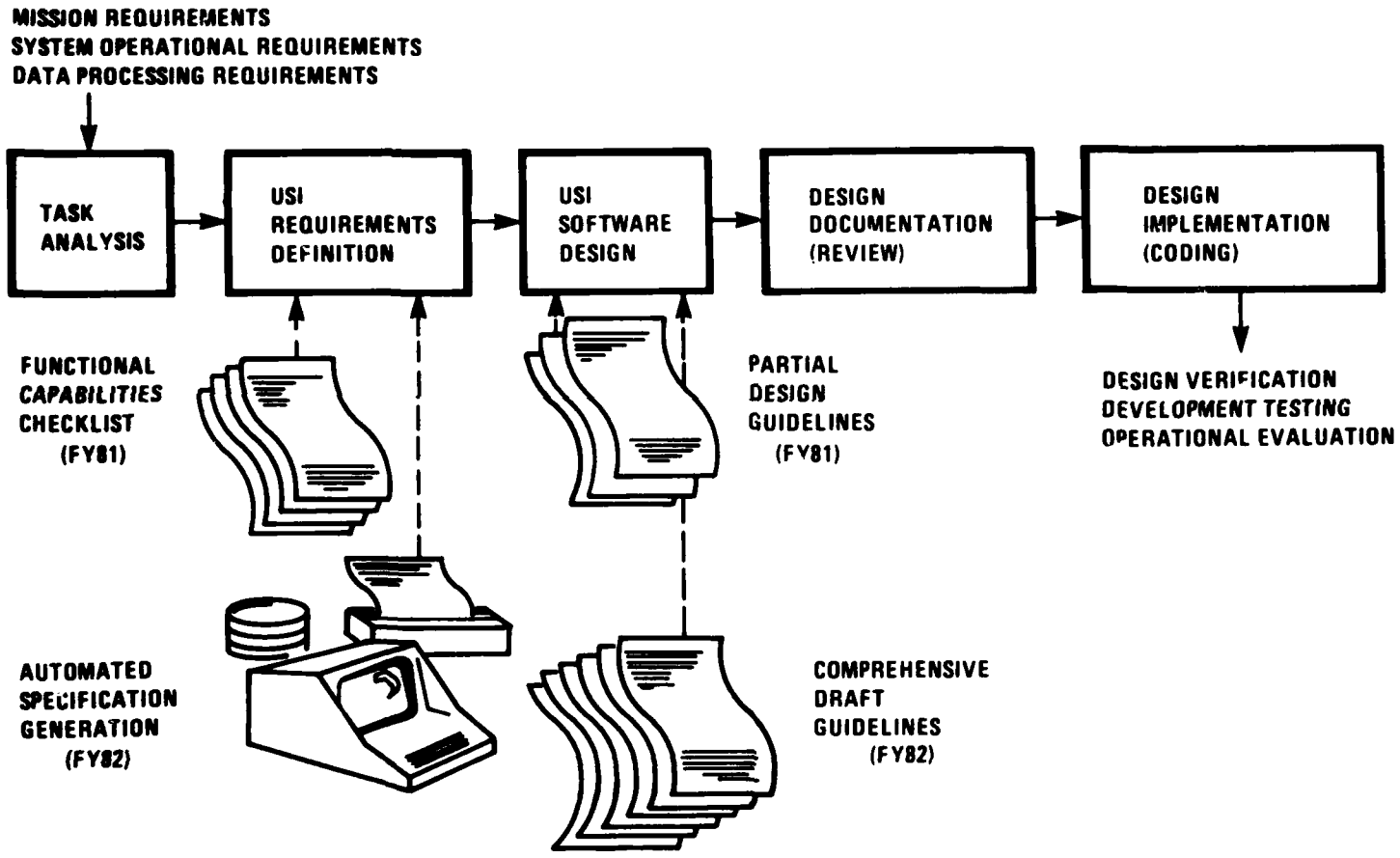


Figure 2. Steps in the USI Design Process

transaction will imply the need for certain functional capabilities at the user interface, i.e., specific features to be provided in USI design.

How many different kinds of user data handling tasks are there? At the most detailed level of analysis, every task is potentially different from an infinite number of others. Considered more generally, however, tasks of a particular type may seem so similar as to justify describing them as variations of a single characteristic task ("generic task", "archetype", etc.). It seems probable that if such characteristic tasks could be identified, some finite number -- perhaps 50 or 100 -- would provide a fair representation of the kinds of tasks users actually perform in information systems. If the functional capabilities required by each of these characteristic tasks were defined and catalogued, that would contribute significantly to good USI design.

Potential confusion in the terminology of task analysis sometimes results from failure to distinguish between functional data handling tasks and user jobs. What about jobs, where do they fit in?

To define a job, tasks must be allocated to determine a set of responsibilities and activities for each system user. Task allocation may not prove easy. It will seldom be desirable to allocate this task to a person and that task to a machine, as textbooks suggest. Instead, more effective performance may result when the computer and its users participate jointly in task performance (Ramsey and Atwood, 1979).

If this is the case, then job definition involves specification of user participation in tasks. From this viewpoint, one possible contribution of USI guidelines would be to suggest the most effective ways in which people can participate in various identified data handling tasks, as well as to specify the functional USI capabilities needed to facilitate such participation.

It should be noted that jobs, as defined here, are generally not the same as system data processing functions. Jobs can be both smaller and larger than functions. A surveillance function, for example, might involve a monitoring task performed by several different operators, perhaps each responsible for a different surveillance sector. And each of those operators, as part of his job, might also participate in other tasks related to different functions.

Thus job definition, the combining of tasks performed by people, involves something more than functional analysis, an extra

application of judgment in the design process. The purpose of job definition is twofold. On the one hand, the definition of jobs determines what capabilities will be required of system users. On the other, the definition of jobs indicates what tasks may have to be performed in combination, by a particular person at a particular work station, and so implies what combination of functional capabilities will be required of the USI.

FUNCTIONAL CAPABILITIES CHECKLIST

Task analysis will not lead directly to USI design. As indicated in Figure 2, an intervening step is needed, where the results of task analysis are defined in terms of required USI functional capabilities. What tools can be provided to aid in USI requirements definition? That is the question discussed next.

If a user must interact with a graphic display, as in a computer-aided design task, he will need a functional capability for pointing at different parts of the display. Without a pointer to designate displayed objects and positions, task transactions would be difficult or even impossible to accomplish. Various physical means might be chosen to provide a pointer -- a lightpen, perhaps, or joystick, or trackball, or whatever. But it is the functional capability for pointing that is required.

For an on-line editing task, where the user must designate arbitrary portions of displayed text for correction, a pointer would certainly seem useful if not essential. For a task involving sequential selections among computer-displayed options, a pointer would probably be useful, although other design alternatives such as multifunction keys might do as well. For many other tasks a pointer might not be needed at all.

Two ideas are illustrated here. First, it may be possible to define the functional capabilities required by a task without immediate regard for their implementation in USI design. Second, different tasks will require different capabilities.

There are many functional capabilities, in addition to the "pointing" example above, that must be considered in USI design. An initial attempt at listing USI functional capabilities included over 200 items (Smith, 1980a). A more recent listing was revised and enlarged to include 473 items (Smith, 1981a).

The current list of USI functional capabilities is presented in Appendix A to this report. This list will continue to grow as more system applications are analyzed, each emphasizing particular USI

requirements with consequent elaboration of that portion of the list. Eventually the list should reach a stable size of perhaps 1000 entries.

This list has been hierarchically structured into six major functional areas -- data entry, data display, sequence control, user guidance, data transfer and data protection. Within each functional area, more specifically identified functions are listed. Under data entry, for example, the specific functions include position designation, direction designation, text entry, data forms, etc.

Each of these specific functions is further divided into subordinate topics of increasing specificity, ranging to seven levels deep in some portions of the list. In this list, for example, the function of position designation, which is the "pointing" function mentioned earlier, is elaborated in its various aspects, to help define just what kinds of pointing may be required for any user task.

Insofar as possible, the checklist is structured and worded to specify functional requirements independently of design implementation. Some functional capabilities, such as pointing, may imply the need for special interface hardware. Many functional capabilities of the USI, however, have little to do with hardware selection, and depend primarily on the logic of software implementation.

Consider two examples. For many data entry tasks users will need a flexible capability to back up and change previously entered items. For a particular data entry task, a user may need the capability to defer entry of unavailable data items, even though they are usually required for subsequent data processing calculations. Such capabilities can be provided only by appropriate USI software design.

The software designs chosen to implement these two functional capabilities -- optional backup and deferred data entry -- might share some common features, e.g., some sort of suspense file. Software implementations would differ in other respects, e.g., in terms of validation checks applied during data entry and at later stages of data processing, and the messages displayed to the user when data deficiencies are detected. Thus the definition of USI functional requirements may influence the design of internal computer processing as well as the design of software directly implementing the user interface.

Given an extended list of USI functional capabilities, one could check each item to record an estimate of whether it is

required, potentially useful, or not needed for performance of a particular task. It is a "checklist" format that is shown in Appendix A. Such a checklist will then define a "profile" of USI functional requirements for the particular task being considered.

If the checklist were completed for each of the various tasks comprising a user's job, and if this were done for all jobs in a system, then the summation of those checklists would define the overall USI functional requirements for that information system design.

In actual practice, the checklist might not be used in this way, repetitively for each task. Instead, a system analyst might simply make one aggregate list to summarize general understanding of a user's job, particularly in early stages of system development when detailed task analyses have not been completed. Practical use of the checklist is discussed further in the remainder of this section.

USE OF THE CHECKLIST

How useful will the checklist be in system development? At this stage it is too soon to tell. Certainly something seems needed. In the survey cited earlier (Smith, 1981b), most respondents, 78 percent, believe that the proposed USI functional capabilities checklist could prove helpful in requirements definition.

If nothing else, the checklist can help ensure comprehensive review of USI functions by system developers. In this regard, one could imagine that similar checklists dealing with other major system functions -- sensors, communication, computer security, etc. -- might also prove helpful in system requirements definition.

Another value of the checklist can be to help system users participate in the early definition of required functional capabilities. Users are often asked what features they want included in a new system, but seldom know in what terms they should reply. The checklist could clarify the expression of user judgment in generating and reviewing system specifications.

To assess checklist use, it will be necessary to attempt its application in a variety of system development programs. The result should be revision and enlargement of the checklist to provide more complete coverage, plus increased understanding of how and when the checklist can best be used.

Looking ahead, two improvements in checklist application seem likely. First, as experience is gained in use of the checklist, it should be possible to accumulate checklists as they are prepared for various characteristic user tasks, so that they become an available reference in the development of new systems. This accumulation of past wisdom can provide a structured embodiment of "corporate memory" in a form potentially valuable to system developers. Then each new USI design will not have to start from scratch.

Second, it should be possible to implement the checklist as an on-line computer aid. That would offer several advantages. The short, somewhat cryptic labels in the checklist could be supplemented by optional display of explanations and examples. Checking a particular capability as not needed could result in automatic omission of all related sub-categories, thus by-passing portions of the (lengthy) checklist. Once required functional capabilities had been checked, it might prove feasible to translate those checks automatically into a verbal description of USI requirements, which could constitute a first draft for that portion of the written system specification. Experiments toward that end are currently under way at MITRE, as discussed in Section 5 of this report.

How is the checklist related to USI design guidelines? An important use of requirements definition will be to limit the choice of design guidelines to be employed during system development. As discussed later in this report, a total catalog of design guidelines might grow quite large, including hundreds of entries. But many of these guidelines will be specifically related to particular functional capabilities. Thus when the subset of capabilities required by a new system has been determined, then a related subset of guidelines could be selected, tailored to system requirements.

If the functional capabilities checklist were automated, as envisioned above, then it should be possible to provide computer aids to facilitate this tailoring process. If both the checklist and the design guidelines were represented in computer storage, with appropriate cross indexing, then specification of desired functional capabilities could be programmed to produce automatic printout of the corresponding set of guidelines for USI design.

SECTION 4

USI DESIGN GUIDELINES

Once USI functional requirements have been defined, the next step is to provide design guidance for the USI software needed to implement those requirements. Current guidelines are scattered in various publications, and in aggregate do not provide comprehensive guidance for USI design. Some general principles can be offered to aid design in major USI functional areas such as data entry, data display, or sequence control. But it will be necessary to develop a coherent structure of detailed guidelines related to specific USI functions before a comprehensive design standard can be established.

CURRENT STATUS

Until recently, there has been no thorough-going attempt to integrate the scattered papers, articles and technical reports that constitute the literature of user-computer interaction. A first step was made, under sponsorship of the Office of Naval Research, in compilation of an extensive bibliography on this subject (Ramsey, Atwood and Kirshbaum, 1978). A significant follow-on effort culminated in publication by Ramsey and Atwood (1979) of a comprehensive summary of this literature. In that compendium the authors characterize the current unsatisfactory situation with regard to USI design guidance:

> In some well established research areas, such as keyboard design and certain physical properties of displays, guidelines exist which are reasonably good and fairly detailed. Such guidelines may be quite helpful in the design of a console or other interface device for a system, or even in the selection of an appropriate off-the-shelf input/output device. As we progress toward the more central issues in interactive systems, such as their basic informational properties, user aids, and dialogue methods, available guidelines become sketchy and eventually nonexistent. The interactive system designer is given little human factors guidance with respect to the most basic design decisions. In fact, the areas in which existing guidelines concentrate are often not even under the control of the designer, who may have more freedom with respect to dialogue and problem-solving aids than with respect to terminal design or selection.
>
> (Ramsey and Atwood, 1979, page 2)

Most published reports dealing with the user-computer interface describe applications rather than design principles. The bibliography of literature on human factors in computer systems, cited above, includes 564 items, but identifies only 17 as offering design guidelines (Ramsey, et al., 1978). A popular book on the design of user-computer dialogues offers stimulating examples, covering a range of on-line applications, but is disappointing in its failure to emphasize design principles (Martin, 1973).

Although accepted standards for USI design are not now available, some first steps toward guidelines development have been taken. In the past decade, as increasing experience has been gained in the use of on-line computer systems, a number of experts have attempted to set forth principles ("guidelines", "ground rules", "rules of thumb") for design of the user-computer interface. None of these sets of guidelines is sufficiently comprehensive to constitute a USI design standard, but they do offer a foundation on which to build.

Looking at the available literature, some published guidelines are proposed in very general terms: "know the user" (Hansen, 1971). Some emphasize special aspects of interface design: response time (Miller, 1968); error protection (Wasserman, 1973); command languages (Kennedy, 1974); multifunction switches (Calhoun, 1978); lightpen selection (Uber, Williams and Hisey, 1968); graphic interaction (Foley and Wallace, 1974); display formatting (Stewart, 1976; Green, 1976); color coding (Krebs, 1978).

Some guidelines have been proposed for specific operator tasks, such as document retrieval (Thompson, 1971), or process control (Williams, 1977). Some have been oriented toward the general use of on-line systems, with little task specificity (Nickerson and Pew, 1971; Palme, 1975; Chariton, 1976; Dzida, Herda and Itzfeldt, 1978). Some guidelines are based on explicit assumptions of system architecture and equipment capability (Pew and Rollins, 1975; see also Pew, Rollins and Williams, 1976). Some guidelines assume implicit constraints, such as printed outputs rather than electronic displays (Chamberlain, 1975).

Only a few sets of guidelines have been expressed at the level of detail needed by designers. Perhaps the most detailed USI guidelines are those proposed by Engel and Granda (1975). Here is a sample: "If a fixed length word or collection of characters is to be entered via the keyboard, limit the field on the screen by special characters, for example, underscores." More specific guidelines of this sort are needed.

In aggregate, the evidence of concern for USI design principles is encouraging, but there is still much to be learned. The current Army effort to develop USI design guidelines, cited earlier, could contribute significant further knowledge of this subject, if its results can be generalized beyond the restrictions on user population and data handling tasks adopted for that study (ARI, 1979).

Military agencies, of course, are not the only organizations concerned with USI design. There is increasing interest in this topic within industrial and commercial organizations, and throughout the general community of people who develop and use information systems. At one recent meeting, over 200 people attended a session of papers on USI design (Ramsey and Atwood, 1980; Granda, 1980; Smith, 1980b; Sidorsky and Parrish, 1980; Pew, Sidner and Vittal, 1980). A questionnaire follow-up to people expressing interest in USI design has resulted in approximately 80 responses and the formation of an informal USI guidelines group (USIGG) of people wishing to exchange information on this topic (Smith, 1981b).

David Penniman, writing for the User On-Line Interaction Group of the American Society for Information Sciences, cites the need for "an interim set of guidelines for user interface design based on available literature and pending the development of better guidelines as our knowledge increases" (1979, page 2). Penniman goes on to remind us that interim guidelines are better than no guidelines at all.

Respondents to the USIGG survey support Penniman's activist position. Given a choice between trying to develop a complete set of USI guidelines now, when many of them must be based on judgment rather than experimental data, or else accepting only a partial set of guidelines based on evaluated research, most respondents, 73 percent, would go with judgment now. Only 14 percent prefer to wait for data later.

It is clear, of course, that system developers cannot wait for future research data in making present design decisions. To meet current needs, a number of in-house handbooks have recently been published to guide USI design within particular organizations (NASA, 1979; Galitz, 1980; Brown, Burkleo, Mangelsdorf, Olsen and Williams, 1980). These in-house guidelines draw heavily from earlier publications, especially the IBM report by Engel and Granda (1975), usually modified by the authors' own good judgment. They will help system developers until such time as a comprehensive USI design standard becomes available.

To assess the value of USI design guidelines, particularly those based on judgment, it will be necessary to try them out in actual system development programs. That process of guidelines evaluation will have to extend over a period of some years. Meanwhile, whatever guidelines are proposed at this stage can only be regarded as tentative, to be used on an interim basis until experience is gained in their trial application.

An extended guidelines development and evaluation effort may well be justified, however, in view of the potential long-term benefits. There is one important factor to consider in this regard, which is the relative stability of human engineering guidelines. Design standards for hardware interfaces may change as each new generation of equipment becomes available. But people change relatively little from one generation to the next, in terms of their basic information processing capabilities and limitations.

This relative invariance of people with respect to technology poses both problems and advantages. The most significant advantage is this: if USI guidelines can be expressed in terms of basic human capabilities rather than transient technology, a design standard of enduring value will be established. It may be true that such guidelines can be established only slowly, but that is all the more reason to make a start now.

GUIDELINES COMPILATION

As a first step toward developing needed guidance, currently available guidelines must be compiled and organized into some coherent structure. The organizational structure recommended here is one related to USI functional capabilities, as categorized in the requirements checklist discussed previously. Such a functional organization for USI design guidelines is illustrated in appendices to this report.

In Appendix B, 98 guidelines have been compiled for the various functions and subfunctions of data entry. In Appendix C, 134 guidelines are presented for data display functions. In Appendix D, 143 guidelines are presented for sequence control functions.

How many guidelines might there be altogether? It seems probable that a comprehensive listing of guidelines for all functional areas of USI design might run to over 1000 items, if guidelines are stated at the level of detail illustrated here. That would constitute a formidable compendium. It might be well, then, to consider just what level of detail is needed.

Each appendix begins with several pages of general discussion of design principles. General principles, however, can only guide the designer a little way. Exhortations to provide flexibility, to preserve context, to ensure compatibility between input and output, are not enough. More specific guidance is needed, and the guidelines that follow the introductory discussion are couched in much more specific terms.

No matter how specifically guidelines are stated, however, they will still be subject to misinterpretation. Various detailed features of wording and format will affect the understanding and use of design guidelines (Rogers and Pegden, 1977). For that reason, therefore, the organization adopted in this report for presentation of guidelines deserves some further description.

Each guideline is stated as a single sentence. In some instances a stated guideline is amplified by one or more examples. Where validity of a guideline is contingent upon special factors, exceptions may be noted. Where further clarification seems needed, further comments may be added. Where a guideline is derived from a particular source, a reference note is added. Finally, where guidelines are specifically related to one another, direction to those other guidelines is given.

One consequence of this notation is that a guideline that began as a simple statement can end up filling half a page. Looking ahead, as future knowledge increases, it may prove desirable to annotate guidelines still further. It should be possible, for example, to estimate the relative importance of different guidelines, so that the USI designer can assess the potential risk of ignoring them, or the potential value of compliance.

As the aggregate bulk of material increases, efficient organization of guidelines will be critical. The organization illustrated here is a simple listing in which guidelines are numbered to correspond to the numbered hierarchy of the functional capabilities checklist. This organization of guidelines based on a functional structure offers a significant advantage in the development of guidelines: it shows where guidelines are missing, and so where further guidelines are needed.

For USI designers, the organization of guidelines in terms of functional capabilities will prove particularly useful. As a designer considers each specified function, pertinent guidelines can be readily referenced. Conversely, the designer can quickly determine where guidance is not available, or is not relevant to functional requirements.

FUTURE DEVELOPMENT

Current efforts at MITRE are directed toward continuing to revise and expand the compilation of USI design guidelines presented here. Next year should see completion of initial draft guidelines for all six functional areas of USI design, adding guidelines for user guidance, communication and security functions.

With a comprehensive compilation of available guidelines, a good start will have been made toward the establishment of USI design standards, but much more must still be accomplished. Compiled guidelines must be reviewed, revised, expanded, applied and evaluated before they become accepted and codified as design standards. Some of the possible steps needed for future development of USI design guidelines are illustrated in Figure 3.

Informal review of compiled guidelines is being solicited through USIGG and other professional groups, in the expectation that guidelines can be improved through additions and changes. Trial application of USI design guidelines is being explored in several Air Force system acquisition programs. Evaluation of proposed guidelines, by system developers, designers and users, still lies in the future.

Once a comprehensive set of guidelines has been prepared, the appropriate form for initial codification will probably be as a design handbook. Such a handbook could be referenced for optional design guidance in system development, or perhaps adopted by some organizations as an agreed standard for in-house system design. With increased experience in its use, a USI design handbook might become formally accepted as a more general design standard, to be referenced during system development for contractual purposes. As a formal standard, guidelines can be referenced for design evaluation as well as for design guidance.

The potential value of design standards is generally recognized in system engineering, particularly with respect to interface design where communication standards are needed to ensure the effective linking of systems, and of system components. USI design standards may prove equally important for the effective linking of a system with its users.

Problems will undoubtedly arise in the application of USI design guidelines. The sheer size of a design handbook, or a documented design standard, may make it difficult for designers to assimilate guidelines, at least on first reading.

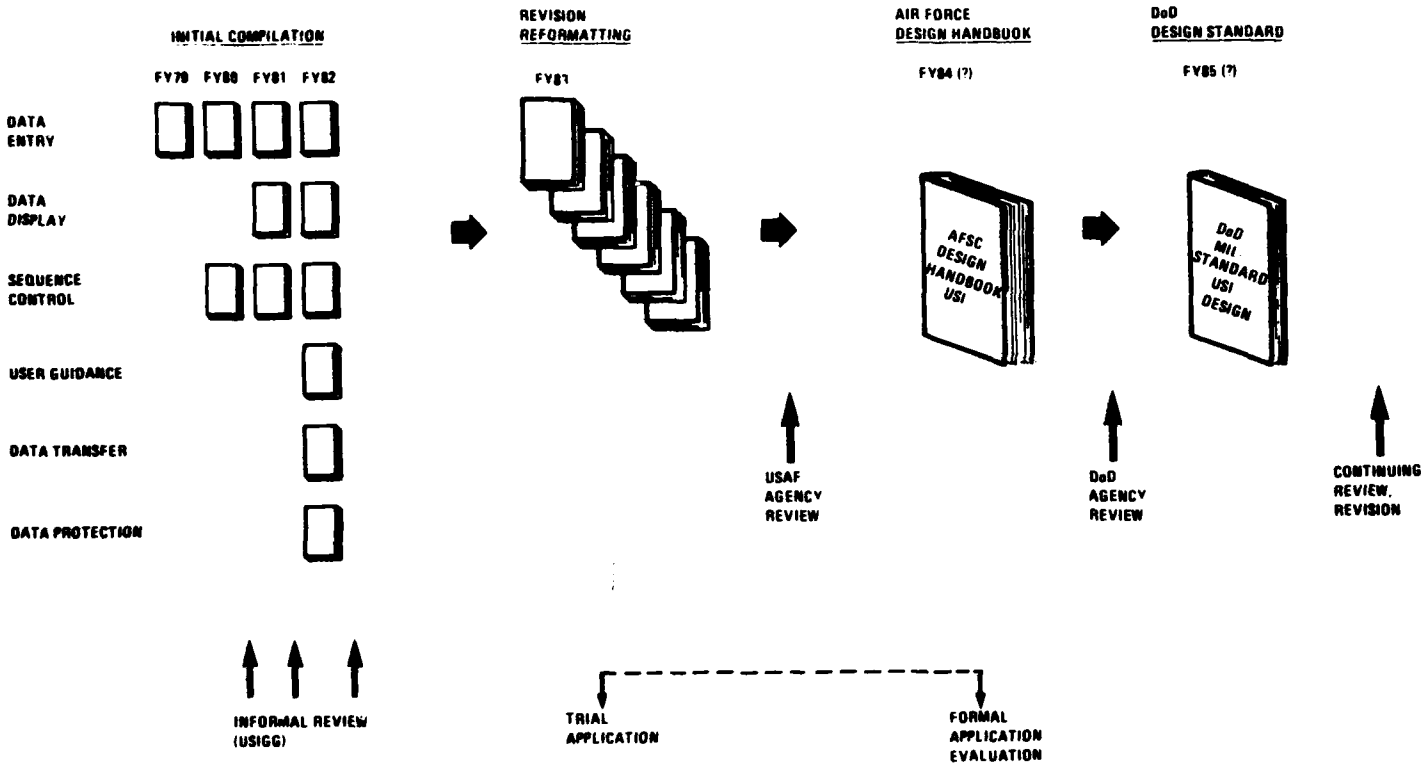


Figure 3. Future USI Guidelines Development

There will also be questions of interpretation, no matter how carefully guidelines are worded. Consider some examples taken from the guidelines for position designation in Appendix B. How can a designer be certain that a cursor is "visually distinctive" (guideline 1.1-2)? What kind of "feedback" should be given a user, and just how fast is "immediate" (1.1-5)? How will a designer without extensive further information know whether controls are "compatible in operation" (1.1-17)?

An accepted standard, then, will not in itself guarantee good design. This is true in the case of human engineering standards for hardware design (Rogers and Armstrong, 1977), and will presumably prove true for future design standards pertaining to USI software. That is no reason to abandon standards. It simply means that something more may be needed.

Two ideas deserve consideration here. First, guidelines and standards will only prove effective if they are applied as part of a broader design methodology. This point has been made by Ramsey and Atwood (1979; 1980). Effective application of USI design guidelines will be dependent upon requirements definition, of the sort recommended in the preceding section of this report. Effective use of guidelines will also depend upon documentation of USI design for review and coordination, as discussed in the next section.

A second idea for ensuring effective use of standards in USI design is more speculative. Once design standards have been established, it may be possible to implement those standards in modular software building blocks that can be used to provide a common, standardized USI design for different systems (Clapp and Hazle, 1978). A practical trial of this approach, for a limited range of data handling applications at IBM, indicates that this is a potential means of achieving significant productivity increases in USI software design (Lee and Santarelli, 1979).

Modular USI software would help ensure uniform implementation of design standards. In effect, once a really good interface has been designed for one system, it may be possible to use it for other systems also. If that <u>is</u> possible, it seems pointless to re-invent a good USI design over and over again.

Future development of modular USI software can bring standardization of functional capabilities, as defined in the checklist, along with standard design implementation. Such a development effort may take a long time to achieve its goal. But with the promise of substantial benefits for system operation as well as for system development, that goal is worth working toward.

SECTION 5

USI DOCUMENTATION

Having defined required USI functional capabilities, and having selected appropriate design guidelines, a third step leading to good USI design is careful design documentation. Current USI documentation in system development is seldom adequate. More complete documentation is needed, to permit review of a proposed USI design before its software implementation, and continuing design coordination thereafter.

CURRENT STATUS

In system development it is customary to require documentation in advance of design, at least for critical design elements. In systems where the USI design is considered critical, which probably includes most information systems, documentation should describe interface software design as well as hardware, including for each on-line transaction the expected user inputs, required data processing, format of displayed outputs, and the associated sequence control logic.

Several questions in the USIGG survey (Smith, 1981b) dealt with USI documentation during system development. Respondents provided estimates of documentation coverage in initial specifications for 20 systems. For five of those systems, there was no documentation of the USI. But that includes two systems for which no specifications at all were written, i.e., where the USI was not slighted in comparison with other aspects of system design.

Some respondents reported modest coverage of the USI, with several pages of system specifications devoted to USI functional requirements, hardware and software design. The median estimate of all respondents was 3-5 pages devoted to each of those three topics.

The most extensive documentation reported was for a graphical display system for process control in a nuclear plant, where it is estimated that 1500 pages of the system specification were devoted to USI functional requirements and another 2000 pages to USI considerations in software design, which the respondent considers adequate for that system. This particular respondent, however, does cite a need for broader system design standardization within the industry.

Only six other respondents judged USI documentation adequate to support effective system development. The remainder, 67 percent of those dealing with this question, consider USI documentation inadequate.

Some allowance must be made for the professional bias of these respondents. Most are involved in human factors engineering, concerned with the problems of designing complex systems for efficient use, and keenly aware of any deficiencies in system development which may lead to non-optimal design. From this point of view, it is not surprising that many are critical of current practices in USI design, including design documentation. Nevertheless, it does seem clear that there are serious deficiencies in USI documentation in many system development programs.

SPECIFICATION GENERATION

An important step in the early conceptual stages of system development is documenting specifications of required functional capabilities. System specifications, if comprehensive, will represent an amalgam of judgment from general system analysts and from specialists in the various functional areas involved in system design -- sensors, communication, data processing, etc. A diverse range of knowledge and experience must be brought to bear in the specification and documentation of system requirements.

In any system engineering group there will be individuals expert in some functional areas but not in others. Ideally, each person in the group can draw upon experience to write a particular portion of the system specification, and the group as a whole can provide the necessary comprehensive coverage.

If there is in the group no expert for a particular functional area, additional help may be needed. One solution is to find an outside expert, who must then learn enough about the proposed system to contribute effectively to its specification.

If an outside expert is not available, an alternative solution might be to use computer aids to help guide the writing of system specifications in any particular functional area. Such computer aids are not presently available, but it seems possible that they could be created.

Such aids could take various forms. At the simplest level, one could store standard paragraphs for relatively fixed material, sometimes called "boiler plate". At a more complicated level, one could store outlines and templates, which the writer of

specifications could refer to, "fill in the blanks", and amplify with additional material specific to a particular system.

At a still more complicated level, one might program a computer to help guide the specification of functional requirements, and to generate draft written material automatically tailored to those requirements. It is this level of specification generation that will be discussed here.

A number of conceptual and procedural problems must be solved in order to develop such a computer aid. The system analyst must have some means of defining system requirements for input to a specification-writing computer. The computer must have a means of generating appropriate written outputs.

The USI checklist presented in Appendix A was designed as a manual aid for requirements definition, to remind a specification writer of topics that should be covered, and to help organize the written specification of USI functional requirements. In recent months, however, the USI checklist, or at least its initial portion dealing with data entry functions, has been re-created as an on-line computer tool.

This automated version of the checklist permits a system analyst to indicate which capabilities are required, or probably useful though not actually required, or not needed. The computer can then look up stored words corresponding to each requirements indicator, and generate a draft specification.

The problem to be solved is how to program a computer to generate an appropriate written output corresponding to any particular pattern of defined functional requirements. In effect, words must be stored in the computer for each item in the checklist, and means devised to link those words together, in any possible combination, to produce a coherent output.

A solution to that problem has been called "patterned prose" (Smith, 1982). In essence, patterned prose consists of a hierarchically related set of sentences, phrases and words, and their logical connectors, organized (and numbered) to correspond to the structure of the checklist which is used to call them out. The words themselves represent a fairly straightforward expansion of the item labels in the checklist. In addition, patterned prose includes various logic elements that serve to connect the words.

A sample of the written output produced by the patterned prose technique is provided in Figure 4. In that sample, it is assumed that <u>every</u> listed capability for data entry has been marked as

*.1.1 Position Designation. A capability for the user to designate positions on a display by cursor placement, and to enter designated positions for computer processing, shall be provided. The user shall be able to place the cursor in arbitrary positions anywhere on a display, by continuous positioning, or at discrete intervals, at least TBD horizontally and TBD vertically. The user shall be able to place the cursor in predefined positions, i.e., at specified display locations, including a HOME position to which the cursor can be returned by a single action from any other point, at the upper left, center, or lower right of the display; a command entry area, where user inputs will be composed; and at the end of a displayed file, where additional user entries will be frequently made. The user shall be able to displace the cursor to positions incremented from a current position by specified intervals, including character spacing right, left, up, and down; spacing by other fixed interval, i.e., tabbing right, left, up, and down; and spacing by format features, including word, line, and paragraph.

*.1.2 Direction Designation. A capability for the user to designate direction, e.g., azimuth, bearing, heading, shall be provided. This shall include a capability for vector rotation on a display; sequential pointing to displayed positions; and numeric entry of directional data.

* Note that the asterisk in these paragraph numbers can be replaced by whatever number(s) the specification writer wishes to designate, so that the draft material will appear properly numbered in the overall specification.

Figure 4. Sample Output of Patterned Prose

required, except for those items labeled "other". The resulting prose does seem to read smoothly, even if it lacks literary merit. Certainly such written output might be considered acceptable as the first draft for a specification.

The automated checklist, generating written specifications derived from patterned prose, is currently available for demonstration at MITRE. Written output can also be displayed on-line for system analysts who may wish to review the results for any defined set of functional requirements before printing a draft specification. This computer tool will soon be offered for evaluation in practical use in MITRE's system acquisition programs.

Looking further ahead, if this approach to specification generation should prove useful in its practical application, then it would probably work for other functional areas as well as for USI functions. That is to say, it may be possible to define requirements and generate written outputs in a similar way for other portions of a system specification.

If that proves true, this technique may represent something rather more than a specification writing machine. It may offer a way to package expertise, to record past experience ("corporate memory") so that it can be applied more effectively in future work. One can imagine expert committees engaged in periodic review to update the structure and wording of computer-stored material in each functional area. And one can imagine that system analysts would no longer need to re-invent the process of specification writing for each system, but instead could rely on past work to help begin new ventures.

DESIGN DOCUMENTATION

Following documentation of system specifications must come documentation of system design. What would constitute adequate documentation of USI design? First, there should be an explicit statement of USI requirements, perhaps using the functional capabilities checklist proposed here, in order to aid mutual understanding among system developers, designers and users. Second, if design guidelines are followed, those should be included or referenced in USI documentation. Third, some effective means is needed to document the design itself, as it evolves during the course of system development.

Documentation of USI design must deal with both content and format. One possible approach is that proposed by Pew and Rollins (1975) to document display formats, data inputs, user actions and

their consequences. Pew and Rollins recommend five components for design documentation. With some changes of terminology, these five components can be summarized as follows.

1. Task Flow Chart. This breaks down a generally stated user task into a logical series of decisions to be made and information handling activities to be performed. Some of these activities will involve interaction with a computer subsystem, and some may not. Task analysis at this level may be provided in a system functional specification, but more often must be developed in the first stage of system design.

2. Transaction Flow Chart. This breaks down those activities involving computer use into a series of discrete transactions. The planned user-system dialogue is outlined step by step, or as is often said for dialogues involving visual displays, frame by frame. The transaction flow chart indicates choices the user can make in sequence control, contingent branching as a result of user input, data entries required, outputs generated, etc.

3. Display (Frame) Format. Working from the transaction flow chart, each step or display frame must be specified in detail. For displays, the most effective way to do this is to design a facsimile showing the exact wording, spacing, etc., that will be used. Here is where USI guidelines could be particularly helpful to the designer.

4. Input Data Definition. For transactions involving data entry, it will be necessary to define each item to be input. This definition will involve specifying the range of acceptable values for each entry where data validation is required, the consequences of data entry in terms of required computer processing, and the implications (if any) for subsequent sequence control and for user guidance.

5. User Guidance. For each transaction the designer should specify the feedback that will result from any possible user action, e.g., alarm or alert signals, guidance messages, other display changes. Often this aspect of USI design is conceived narrowly, to include only the specification of error messages. It will help USI design, however, if the designer can adopt a broader perspective. There are many possible aspects of user guidance that cannot fairly be termed error messages. Perhaps the user should be given a message requesting confirmation of a doubtful entry, i.e., an entry that is so unlikely that it is probably though not certainly an error. Perhaps the user should be requested to enter additional data to amplify an entry just made. Or perhaps the user should simply be given a message suggesting the next step in a transaction sequence.

If all of these five components were included in USI documentation, which is not often the case, then collectively they would seem to provide an adequate basis for design review and coordination. A more recent report, however, suggests that something more may be needed for USI documentation, namely, the development of a "knowledge representation language" that can be used to provide procedural description both in terms of the user's view of the system and the designer's specifications (Pew, et al., 1980). It remains to be demonstrated how such an approach will work in practice.

DESIGN REVIEW

Periodic design review is needed to ensure successful system development. Early review allows critique of inadequate design features before actual implementation, at a stage when suggestions for improvement of USI design do not entail the costs and delays of software modification. To permit effective review, documentation must clearly explain the proposed USI design to both system developers and intended users.

In some system development efforts, the designers may be encouraged to propose specific USI design guidelines not included in the system specification. Guidelines proposed by the designer should be included in USI design documentation, so that design review can determine whether USI functional requirements will be met.

In a development program involving modifications to improve an existing system, rather than development of a completely new system, the designers may be asked to define USI guidelines consonant with current procedures, which represent de facto design standards. With increased emphasis on incremental procurement, the modification and expansion of existing capabilities may become a typical mode of future system development programs.

One aspect of USI design review deserves particular attention. For many systems the USI represents such a significant and pervasive element of the total effort that review of USI design can be a large first step in reviewing overall system design. Where users will be intimately involved in the performance of critical system functions, then a detailed review of proposed user functions will go a long way toward understanding overall system functions. Thus a document describing USI design, with some amplification to include internal data processing and interfaces plus relevant external constraints, could become a description of the larger information system in which the users will work.

Such reference to USI design in order to clarify understanding of more general system operation can be observed in practice. In system development programs USI design documents are sometimes generated informally, outside of contractual requirements, as a spontaneous aid to overall design review and coordination.

Pew and Rollins (1975) recommend that once USI design has been documented it should be subject to configuration management control. That is to say, during software implementation, changes to a documented USI design should be made only after review and authorization, just like other system interfaces. Such control will help ensure design consistency, so that different people can implement different parts of the USI with reasonable confidence that the parts will fit together as a coherent whole. That seems a good idea.

DESIGN COORDINATION

USI documentation can serve a variety of purposes in design coordination. During initial software implementation, a USI "design handbook" could provide a single reference to help coordinate the contributions of different people in hardware and software engineering groups. Such a handbook could also provide the basis for configuration management of USI design, as subsequent changes are proposed to the initial design implementation.

For the purpose of coordination, USI design documentation should include a summary of the guidelines actually used, i.e., as adapted from initial design proposals to take into account subsequently imposed software tradeoffs, hardware constraints, etc. In effect, the USI designer should document his work as if he were telling someone else how to do it.

Such explicit documentation of system-specific USI guidelines is presently rare, but is nonetheless needed. A recent example is provided in a NASA report of USI guidelines providing a sort of design handbook for Spacelab experiments:

> The purpose of this document is to present CRT display design and command usage guidelines applicable to experiments utilizing the Experiment Computer Application Software (ECAS) for use by Spacelab scientific investigators and experiment designers. It is expected that most Spacelab experiments will have need of such displays.

> The guidelines, while not given as strict requirements, explain recommended methods and techniques for presenting data from the ECAS program via the DDS [data display system] to the payload crew. The user is encouraged to apply and follow them, although other considerations (memory conservation, etc.) may force a trade-off in specific instances. Used as a reference, the document will be an important aid in standardizing the crew/experiment interface among the different payloads and should result in lower crew training time and increased efficiency of the payload crew in onboard experiment operations.
>
> (NASA, 1979, page 1-1)

A USI design handbook could provide timely information to coordinate the preparation of user manuals and training materials in parallel with design implementation, throughout the course of system development.

USI design documentation could also continue to provide helpful guidance after system implementation, for users who may wish to add or modify inputs, displays, sequence control logic, etc., in order to meet changing operational requirements. That may become more common in the future, as existing information systems are upgraded rather than new systems built. In such situations, it could be important to maintain consistency in USI design between "old" and "new" portions of a system.

It may seem redundant to keep a written description of USI design once a system is operating. Why not just look at the actual on-line interface to see what it is like? Certainly it is true that operational use can provide valuable insight into the advantages and drawbacks of any particular USI design; and it is operational performance that provides final design verification. But operational use is a slow and inefficient way to review some design features, for example, to discover system response to inputs seldom made and transaction sequence paths seldom taken.

Moreover, the principles underlying USI design, the functional requirements and the chosen guidelines for meeting them, will not all be evident in operational use. It may well be easier for future developers and users if those principles can be set down explicitly for each system, as well as more generally for all systems. Which brings this report full circle in its recommendations: system developers and prospective users should define and document functional USI requirements and guidelines for the designer; and USI designers should document for the system developers and users just what has been done.

Improvements in USI requirements definition, design guidance and design documentation can come only as the result of a continuing collaborative effort by people working on these problems. No one person has a monopoly on wisdom in this area. If an effective joint effort can be sustained through the next decade, significant advances in the art and methodology of USI design will be achieved. You can join in that effort.

REFERENCES

ARI (Army Research Institute for Behavioral and Social Sciences). Development of Design Guidelines and Criteria for User/Operator Transactions with Battlefield Automated Systems, RFP No. MDA903-79-R-0218, 1979.

Brown, C. M., Burkleo, H. V., Mangelsdorf, J. E., Olsen, R. A., and Williams, A. R., Jr. Human Factors Engineering Criteria for Information Processing Systems. Sunnyvale, California: Lockheed Missiles and Space Company, 10 October 1980.

Calhoun, G. L. Control logic design criteria for multifunction switching devices. In Proceedings of the 22nd Annual Meeting. Santa Monica, California: Human Factors Society, 1978, 383-387.

Chamberlain, R. G. Conventions for interactive computer programs. Interfaces, November 1975, 6(1), 77-82.

Chariton, D. R. Man-machine interface design for timesharing systems. In Proceedings of the Annual Conference. New York: Association for Computing Machinery, 1976, 362-366.

Clapp, J. A. and Hazle, M. Building Blocks for C3 Systems, Report ESD-TR-77-360. Bedford, Massachusetts: USAF Electronic Systems Division, March 1978. (NTIS No. AD A052 568)

Dzida, W., Herda, S. and Itzfeldt, W. D. User-perceived quality of interactive systems. In Proceedings of the 3rd International Conference on Software Engineering, IEEE Catalog No. 78CH1317-7C, 1978, 188-195.

Engel, S. E. and Granda, R. E. Guidelines for Man/Display Interfaces, Technical Report TR 00.2720. Poughkeepsie, New York: IBM, December 1975.

Foley, J. D. and Wallace, V. L. The art of natural graphic man-machine conversation. In Proceedings of the IEEE, April 1974, 62(4), 462-471.

Galitz, W. O. Human Factors in Office Automation. Atlanta, Georgia: Life Office Management Association, Inc., 1980.

Goodwin, N. C. Man-machine interface characteristics. Published as an Appendix in J. A. Clapp and M. Hazle, Building Blocks for C3 Systems, Report ESD-TR-77-360. Bedford, Massachusetts: USAF Electronic Systems Division, March 1978. (NTIS No. AD A052 568)

Granda, R. E. Man/machine design guidelines for the use of screen display terminals. In Proceedings of the 24th Annual Meeting. Santa Monica, California: Human Factors Society, 1980, 90-92.

Green, E. E. Message design -- graphic display strategies for instruction. In Proceedings of the Annual Conference, New York: Association for Computing Machinery, 1976, 144-148.

Hansen, W. J. User engineering principles for interactive systems. In AFIPS Conference Proceedings, 1971, 39, 523-532.

Kennedy, T. C. S. The design of interactive procedures for man-machine communication. International Journal of Man-Machine Studies, 1974, 6, 309-334.

Krebs, M. J. Design principles for the use of color in displays. In SID International Symposium Digest of Papers. Los Angeles: Society for Information Display, 1978, 28-29.

Lee, W. J. and Santarelli, F. O. Sequoyah -- A total application development and delivery system. In Proceedings of the Application Development Symposium (Monterey, October 1979). Chicago: SHARE, Inc. and Guide International Corporation, 1979, 9-15.

Martin, J. Design of Man-Computer Dialogues. Englewood Cliffs, New Jersey: Prentice-Hall, 1973.

MIL-H-48655B. Military Specification: Human Engineering Requirements for Military Systems, Equipment and Facilities. Washington: Department of Defense, 31 January 1979.

MIL-STD-454F. Military Standard: Standard General Requirements for Electronic Equipment. Washington: Department of Defense, 15 March 1978.

MIL-STD-483. Military Standard: Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs. Washington: Department of Defense, 31 December 1970.

MIL-STD-1472B. Military Standard: Human Engineering Design Criteria for Military Systems, Equipment and Facilities. Washington: Department of Defense, 31 December 1974.

MIL-STD-1472C. Military Standard: Human Engineering Design Criteria for Military Systems, Equipment and Facilities. Washington: Department of Defense, 2 May 1981.

Miller, R. B. Response time in man-computer conversational transactions. In AFIPS Conference Proceedings, 1968, 33, 267-277.

NASA (National Aeronautics and Space Administration). Spacelab Experiment Computer Application Software (ECAS) Display Design and Command Usage Guidelines, Report MSFC-PROC-711. George C. Marshall Space Flight Center, Alabama, January 1979.

Nickerson, R. S. and Pew, R. W. Oblique steps toward the human-factors engineering of interactive computer systems. Published as an Appendix in M. C. Grignetti, D. C. Miller, R. S. Nickerson and R. W. Pew, Information Processing Models and Computer Aids for Human Performance, Report No. 2190. Cambridge, Massachusetts: Bolt, Beranek and Newman, Inc., June 1971. (NTIS No. AD 732 913)

Palme, J. Interactive Software for Humans, Report FOA-C10029-M3(E5). Stockholm: Swedish National Defense Research Institute, July 1975.

Palme, J. A human-computer interface for non-computer specialists. Software - Practice and Experience, 1979, 9, 741-747.

Parsons, H. M. The scope of human factors in computer-based data processing systems. Human Factors, 1970, 12(2), 165-175.

Penniman, W. D. Past chairman's message. SIG Newsletter No. UOI-10. Washington: American Society for Information Science, May 1979.

Pew, R. W. and Rollins, A. M. Dialog Specification Procedures, Report 3129 (revised). Cambridge, Massachusetts: Bolt Beranek and Newman, 1975.

Pew, R. W., Rollins, A. M. and Williams, G. A. Generic man-computer dialogue specification: an alternative to dialogue specialists. In Proceedings - 6th Congress of the International Ergonomics Association. Santa Monica, California: Human Factors Society, 1976, 251-254.

Pew, R. W., Sidner, C. L. and Vittal, J. J. Man-machine interface design documentation: Representing the user's model of a system. In Proceedings of the 24th Annual Meeting. Santa Monica, California: Human Factors Society, 1980, 103-107.

Ramsey, H. R. and Atwood, M. E. Human Factors in Computer Systems: A Review of the Literature, Technical Report SAI-79-111-DEN. Englewood, Colorado: Science Applications, Inc., September 1979. (NTIS No. AD A075 679)

Ramsey, H. R. and Atwood, M. E. Man-computer interface design guidance: State of the art. In Proceedings of the 24th Annual Meeting. Santa Monica, California: Human Factors Society, 1980, 85-89.

Ramsey, H. R., Atwood, M. E. and Kirshbaum, P. J. A Critically Annotated Bibliography of the Literature of Human Factors in Computer Systems, Technical Report SAI-78-070-DEN. Englewood, Colorado: Science Applications, Inc., May 1978. (NTIS No. AD A058 081)

Rogers, J. G. and Armstrong, R. Use of human engineering standards in design. Human Factors, 19(1), 15-23, 1977.

Rogers, J. G. and Pegden, C. D. Formatting and organization of a human engineering standard. Human Factors, 19(1), 55-61, 1977.

Rouse, W. B. Design of man-computer interfaces for on-line interactive systems. In Proceedings of the IEEE, June 1975, 63(6), 847-857.

Shneiderman, B. Software Psychology: Human Factors in Computer and Information Systems. Cambridge, Massachusetts: Winthrop Publishers, Inc., 1980.

Sidorsky, R. C. and Parrish, R. N. Guidelines and criteria for human-computer interface design of battlefield automated systems. In Proceedings of the 24th Annual Meeting. Santa Monica, California: Human Factors Society, 1980, 98-102.

Smith, S. L. Man-computer information transfer. In Howard, J. H. (Ed.) Electronic Information Display Systems, 284-299. Washington: Spartan Books, 1963.

Smith, S. L. MAC Air Cargo Data Entry: 1. Preliminary Analysis of Alternative Modes, Technical Report ESD-TR-76-162, Vol 1. Bedford, Massachusetts: USAF Electronic Systems Division, 1 July 1975. (NTIS AD A029 013)

Smith, S. L. MAC Air Cargo Data Entry: 5. Field Testing an On-Line Terminal at the Dover Aerial Port, Technical Report MTR-3066-5. Bedford, Massachusetts: The MITRE Corporation, 30 June 1976.

Smith, S. L. Requirements definition and design guidelines for the man-machine interface in C3 system acquisition, Technical Report ESD-TR-80-122. Bedford, Massachusetts: USAF Electronic Systems Division, June 1980. (a) (NTIS No. AD A087 258)

Smith, S. L. Man-machine interface requirements definition: Task demands and functional capabilities. In Proceedings of the 24th Annual Meeting. Santa Monica, California: Human Factors Society, 1980, 93-97. (b)

Smith, S. L. Man-Machine Interface (MMI) Requirements Definition and Design Guidelines: A Progress Report, Technical Report ESD-TR-81-113. Bedford, Massachusetts: USAF Electronic Systems Division, February 1981. (a) (NTIS No. AD A096 705)

Smith, S. L. Design guidelines for the user-system interface of on-line computer systems: A status report. In Proceedings of the 25th Annual Meeting. Santa Monica, California: Human Factors Society, 1981, 509-512. (b)

Smith, S. L. Patterned prose for automatic specification generation. Paper presented at Conference on Human Factors in Computer Systems, Gaithersburg, Maryland, March 1982.

Stewart, T. F. M. Displays and the software interface. Applied Ergonomics, 1976, 7(3), 137-146.

Thompson, D. A. Interface design for an interactive information retrieval system: a literature survey and a research system description. Journal of the American Society for Information Science, 1971, 22, 361-373.

Uber, G. T., Williams, P. E. and Hisey, B. L. The organization and formatting of hierarchical displays for the on-line input of data. In AFIPS Conference Proceedings, 1968, 33, 219-226.

Wasserman, A. I. The design of 'idiot-proof' interactive programs. In AFIPS Conference Proceedings, 1973, 42, M34-M38.

Williams, T. J. (Ed.) Guidelines for the Design of Man/Machine Interfaces for Process Control. West Lafayette, Indiana: Purdue University, January 1977. (NTIS No. AD A036 457)

Woodson, W. E. Human Factors Design Handbook. New York: McGraw-Hill, 1981.

APPENDIX A

CHECKLIST OF USI FUNCTIONAL CAPABILITIES

USI Capability	Requirement Estimate* R	U	N	
1.0 DATA ENTRY				
1.1. Position Designation				
1. arbitrary positions	.	.	.	
1 continuous	—	—	—	
2 discrete	—	—	—	
2. predefined positions	.	.	.	
1. HOME	.	.	.	
1 upper left	—	—	—	
2 center	—	—	—	
3 lower right	—	—	—	
4 [other]	—	—	—	_____
2 command entry area	—	—	—	
3 end of file	—	—	—	
4 [other]	—	—	—	_____
3. incremental positions	.	.	.	
1. by character	.	.	.	
1 right	—	—	—	
2 left	—	—	—	
3 up	—	—	—	
4 down	—	—	—	
2. by interval	.	.	.	
1 right	—	—	—	
2 left	—	—	—	
3 up	—	—	—	
4 down	—	—	—	
3. by format features	.	.	.	
1 word	—	—	—	
2 line	—	—	—	
3 paragraph	—	—	—	
4 [other]	—	—	—	_____
1.2. Direction Designation				
1 vector rotation	—	—	—	
2 sequential pointing	—	—	—	
3 numeric entry	—	—	—	
4 [other]	—	—	—	_____

---

* R = Required, U = Useful, N = Not Needed

USI Capability	R	U	N	
1.3. Text				
1. format	.	.	.	
1. predefined	.	.	.	
1 header	—	—	—	
2 paragraph	—	—	—	
3 page	—	—	—	
4 [other]	—	—	—	_____
2. user-defined	—	—	—	
2. enter	.	.	.	
1 insert	—	—	—	
2 append	—	—	—	
3 change	—	—	—	
4. delete	.	.	.	
1 character	—	—	—	
2 word	—	—	—	
3 line	—	—	—	
4 sentence	—	—	—	
5 paragraph	—	—	—	
6 page	—	—	—	
5 [other]	—	—	—	_____
1.4. Data Forms				
1. format	.	.	.	
1. predefined	.	.	.	
1 selection automatic	—	—	—	
2 selection by request	—	—	—	
2 user-defined	—	—	—	
2 enter	—	—	—	
3 change	—	—	—	
4. delete	.	.	.	
1 character	—	—	—	
2 field	—	—	—	
3 section	—	—	—	
4 page	—	—	—	
5 [other]	—	—	—	_____

USI Capability	R	U	N	
1.5. Tabular Data				
1. format	.	.	.	
1. predefined	.	.	.	
1 header	—	—	—	
2 row (object)	—	—	—	
3 column (property)	—	—	—	
4 page	—	—	—	
5 [other]	—	—	—	_____
2 user-defined	—	—	—	
2 enter	—	—	—	
3 change	—	—	—	
4. delete	.	.	.	
1 character	—	—	—	
2 field	—	—	—	
3 row	—	—	—	
4 column	—	—	—	
5 page	—	—	—	
6 [other]	—	—	—	_____
1.6. Graphic Data				
1. format	.	.	.	
1. predefined	.	.	.	
1. plot type	.	.	.	
1 geographic	—	—	—	
2 line graph	—	—	—	
3 bar graph	—	—	—	
4 pie chart	—	—	—	
5 [other]	—	—	—	_____
2 background/map	—	—	—	
3 data category	—	—	—	
4 symbol	—	—	—	
5 [other]	—	—	—	_____
2 user-defined	—	—	—	
2 enter	—	—	—	
3 change	—	—	—	
4. delete	.	.	.	
1 point	—	—	—	
2 symbol	—	—	—	
3 drawn line	—	—	—	
4 data category	—	—	—	
5 [other]	—	—	—	_____

USI Capability	R	U	N	
1.7. Data Validation				
1. required entry	.	.	.	
1 immediate	—	—	—	
2 deferrable	—	—	—	
2. length of entry	.	.	.	
1 fixed	—	—	—	
2 maximum	—	—	—	
3 minimum	—	—	—	
3. content of entry	.	.	.	
1 numeric	—	—	—	
2 alphabetic	—	—	—	
3 alphanumeric	—	—	—	
4 defined codes	—	—	—	
5 [other]	—	—	—	_____
4. comparative checks	.	.	.	
1 equal to	—	—	—	
2 greater than	—	—	—	
3 less than	—	—	—	
4 IF...THEN	—	—	—	
5 [other]	—	—	—	_____
5. default entry	.	.	.	
1 predefined	—	—	—	
2 user-defined	—	—	—	
1.8. Other Data Processing				
1. file management	.	.	.	
1 merging/linking	—	—	—	
2. cross-file update	.	.	.	
1 automatic	—	—	—	
2 by request	—	—	—	
2. derived statistics	.	.	.	
1 total	—	—	—	
2 mean	—	—	—	
3 median	—	—	—	
4 range (of values)	—	—	—	
5 [other]	—	—	—	_____
3. other computation	.	.	.	
1 date/time	—	—	—	
2 grid conversion	—	—	—	
3 azimuth	—	—	—	
4 range (distance)	—	—	—	
5 [other]	—	—	—	_____

USI Capability	R	U	N	
1.9. Design Change				
1. type	.	.	.	
1 file formats	—	—	—	
2 entry formats	—	—	—	
3 item specification	—	—	—	
4 data processing	—	—	—	
5 [other]	—	—	—	_____
2. implementation	.	.	.	
1. on-line	.	.	.	
1 by transaction	—	—	—	
2 by software change	—	—	—	
2 off-line	—	—	—	

USI Capability	R	U	N	
2.0 DATA DISPLAY				
2.1. Data Type				
1. text	.	.	.	
1. formatted	.	.	.	
1 messages	—	—	—	
2 reports	—	—	—	
2 unformatted	—	—	—	
2 data forms	—	—	—	
3 tables	—	—	—	
4. graphics	.	.	.	
1 geographic plot	—	—	—	
2 line graph	—	—	—	
3 bar graph	—	—	—	
4 pie chart	—	—	—	
5 background/map	—	—	—	
6 [other]	—	—	—	_____
5 combination	—	—	—	
2.2. Data Selection				
1. automatic	.	.	.	
1 predefined	—	—	—	
2 user-defined	—	—	—	
2. by request	.	.	.	
1 file	—	—	—	
2 file subset	—	—	—	
3 data item	—	—	—	
4. data category	.	.	.	
1 time period	—	—	—	
2 area	—	—	—	
3 [other]	—	—	—	_____
5 combination	—	—	—	
2.3. Data Aggregation				
1 summary display	—	—	—	
2 grouped data	—	—	—	
3 individual items	—	—	—	
2.4. Display Generation				
1. automatic	.	.	.	
1 predefined	—	—	—	
2 user-defined	—	—	—	
2 by request	—	—	—	

USI Capability	R	U	N	
2.5. Display Partitioning	.	.	.	
1. fixed windows	.	.	.	
1 display title	—	—	—	
2 page number	—	—	—	
3 date/time group	—	—	—	
4 error messages	—	—	—	
5 command entry area	—	—	—	
6 [other]	—	—	—	_____
2. variable windows	.	.	.	
1 automatic	—	—	—	
2 by request	—	—	—	
3 multiple displays	—	—	—	
4. printout	.	.	.	
1 from display	—	—	—	
2 from files	—	—	—	
3 selected data	—	—	—	
5. auditory display	.	.	.	
1 alerting signals	—	—	—	
2 error messages	—	—	—	
3 advisory information	—	—	—	
2.6. Display Density				
1. text	.	.	.	
1 high (> 1000 char.)	—	—	—	
2 moderate	—	—	—	
3 low (< 600 char.)	—	—	—	
2. data forms	.	.	.	
1 high (> 600 char.)	—	—	—	
2 moderate	—	—	—	
3 low (< 300 char.)	—	—	—	
3. tabular	.	.	.	
1 high (> 600 char.)	—	—	—	
2 moderate	—	—	—	
3 low (< 300 char.)	—	—	—	
4. graphic	.	.	.	
1 high (> 300 char.)	—	—	—	
2 moderate	—	—	—	
3 low (< 100 char.)	—	—	—	

USI Capability	R	U	N
2.7. Display Coding			
1. variables/dimensions	.	.	.
1 many	—	—	—
2 moderate	—	—	—
3 few	—	—	—
2. categories/values	.	.	.
1 many (> 20)	—	—	—
2 moderate (8-20)	—	—	—
3 few (3-7)	—	—	—
4 just two	—	—	—
3. criticality	.	.	.
1 high	—	—	—
2 moderate/low	—	—	—
4. code format	.	.	.
1 predefined	—	—	—
2 user-defined	—	—	—
5. auditory coding	.	.	.
1 voice	—	—	—
2 other signals	—	—	—
2.8. Display Coverage			
1. displacement	.	.	.
1. page	.	.	.
1. stepwise	.	.	.
1 forward	—	—	—
2 back	—	—	—
2 by page number	—	—	—
2. scroll	.	.	.
1 forward	—	—	—
2 back	—	—	—
3 offset	—	—	—
2. expansion	.	.	.
1. discrete increments	.	.	.
1 predefined	—	—	—
2 user-defined	—	—	—
2. continuous	.	.	.
1 zoom in	—	—	—
2 zoom out	—	—	—
3. return/normalize	.	.	.
1 automatic	—	—	—
2 by request	—	—	—

USI Capability	R	U	N	
2.9. Display Update				
1. initiation	.	.	.	
1 automatic	—	—	—	
2 by request	—	—	—	
2. rate	.	.	.	
1 normal	—	—	—	
2. modified	.	.	.	
1 fast	—	—	—	
2 slow	—	—	—	
3 freeze	—	—	—	
2.10. Display Suppression				
1. initiation	.	.	.	
1. automatic	.	.	.	
1. timeout	.	.	.	
1 one step	—	—	—	
2 gradual fading	—	—	—	
2 [other]	—	—	—	_____
2. by request	.	.	.	
1 data item	—	—	—	
2. data category	.	.	.	
1 time period	—	—	—	
2 area	—	—	—	
3 [other]	—	—	—	_____
3 all data (ERASE)	—	—	—	
2. duration	.	.	.	
1 continuing	—	—	—	
2. temporary	.	.	.	
1 automatic	—	—	—	
2 by request	—	—	—	
2.11. Design Change				
1. function	.	.	.	
1 data type	—	—	—	
2 selection	—	—	—	
3 aggregation	—	—	—	
4 display generation	—	—	—	
5 partitioning	—	—	—	
6 density	—	—	—	
7 coding	—	—	—	
8 coverage	—	—	—	
9 update	—	—	—	
10 suppression	—	—	—	
2. implementation	.	.	.	
1. on-line	.	.	.	
1 by transaction	—	—	—	
2 by software change	—	—	—	
2 off-line	—	—	—	

USI Capability	R	U	N
3.0 SEQUENCE CONTROL			
3.1. Dialogue Type			
1 question and answer	—	—	—
2 form filling	—	—	—
3 menu selection	—	—	—
4 function keys	—	—	—
5 command language	—	—	—
6 query language	—	—	—
7 natural language	—	—	—
8 graphic interaction	—	—	—
3.2. Transaction Selection			
1 general OPTIONS	—	—	—
2 implicit options	—	—	—
3. step-specific options	.	.	.
1 automatic	—	—	—
2 by request	—	—	—
4 stacked commands	—	—	—
5. linked commands (macros)	.	.	.
1 predefined	—	—	—
2 user-defined	—	—	—
3.3. Interrupt			
1 CANCEL	—	—	—
2 BACKUP	—	—	—
3 RESTART	—	—	—
4 ABORT	—	—	—
5 END	—	—	—
3.4. Context Definition			
1 predefined	—	—	—
2. user-defined	.	.	.
1 by command	—	—	—
2 by data category	—	—	—
3 by data item	—	—	—
3.5. Error Management			
1 explicit ENTER	—	—	—
2 automatic validation	—	—	—
3 direct error correction	—	—	—
4 CONFIRM protection	—	—	—

USI Capability	R	U	N	
3.6. Alarms				
1. alarm conditions	.	.	.	
1 predefined	—	—	—	
2. user-defined	.	.	.	
1 variables/dimensions	—	—	—	
2 categories/values	—	—	—	
2. alarm acknowledgment	.	.	.	
1. automatic	.	.	.	
1 by timeout	—	—	—	
2 by implicit action	—	—	—	
3 by correction	—	—	—	
4 by user override	—	—	—	
2. user action	.	.	.	
1 predefined	—	—	—	
2 user-defined	—	—	—	
3.7. Design Change				
1. function	.	.	.	
1 dialogue type	—	—	—	
2 available options	—	—	—	
3 sequence logic	—	—	—	
4 data processing	—	—	—	
5 [other]	—	—	—	_____
2. implementation	.	.	.	
1. on-line	.	.	.	
1 by transaction	—	—	—	
2 by software change	—	—	—	
2 off-line	—	—	—	

USI Capability	R	U	N	
4.0 USER GUIDANCE				
4.1. Status Information				
1. operability	.	.	.	
1 local work station	__	__	__	
2. system	.	.	.	
1 equipment	__	__	__	
2 data files	__	__	__	
3 functions	__	__	__	
3 external	__	__	__	
2 current users	__	__	__	
3 current load	__	__	__	
4 other notices	__	__	__	
5. date/time signals	.	.	.	
1 continuous	__	__	__	
2 periodic	__	__	__	
3 by request	__	__	__	
6. alarm signals	.	.	.	
1 variables/dimensions	__	__	__	
2 categories/values	__	__	__	
4.2. Routine Feedback				
1. *input*	.	.	.	
1 data entry	__	__	__	
2 data change	__	__	__	
3 data deletion	__	__	__	
2. output	.	.	.	
1 data displayed	__	__	__	
2 partial display	__	__	__	
3 data not available	__	__	__	
3. sequence control	.	.	.	
1 requested transaction	__	__	__	
2 changed context	__	__	__	
3 [other]	__	__	__	_____
4.3. Error Feedback				
1 error type	__	__	__	
2 correction procedure	__	__	__	
3 alert signals	__	__	__	
4 cursor position	__	__	__	
5 user confirmation	__	__	__	

USI Capability	R	U	N	
4.4. Job Aids				
1. automatic prompts	.	.	.	
1 fixed messages	—	—	—	
2. contingent on input	.	.	.	
1 data entry	—	—	—	
2 command selection	—	—	—	
3 context change	—	—	—	
3. command aiding	.	.	.	
1 branching options	—	—	—	
2 disambiguation	—	—	—	
3 [other]	—	—	—	_____
4. cursor position	.	.	.	
1 command entry area	—	—	—	
2 data entry field	—	—	—	
3 error location	—	—	—	
4 off screen	—	—	—	
5 [other]	—	—	—	_____
2. by request	.	.	.	
1 data index	—	—	—	
2 command index	—	—	—	
3 HELP, EXPLAIN	—	—	—	
4 on-job training	—	—	—	
5 [other]	—	—	—	_____
3. instructional level	.	.	.	
1 novice users	—	—	—	
2. transitional users	.	.	.	
1 by time of use	—	—	—	
2 by measured skill	—	—	—	
3 [other]	—	—	—	_____
3 expert users	—	—	—	
4.5. User Records				
1 transactions	—	—	—	
2 files accessed	—	—	—	
3 programs used	—	—	—	
4. errors made	.	.	.	
1 data entry/change	—	—	—	
2 sequence control	—	—	—	
5 help requested	—	—	—	
6 [other]	—	—	—	_____

USI Capability	R	U	N	
4.6. Design Change				
1. type	.	.	.	
1 status information	—	—	—	
2 alarms/alerts	—	—	—	
3 error messages	—	—	—	
4 prompts	—	—	—	
5 auxiliary help	—	—	—	
6 training aids	—	—	—	
7 [other]	—	—	—	_____
2. implementation	.	.	.	
1. on-line	.	.	.	
1 by transaction	—	—	—	
2 by software change	—	—	—	
2 off-line	—	—	—	

USI Capability	R	U	N	
5.0 DATA TRANSMISSION				
5.1. Data Type				
1. text	.	.	.	
1. formatted	.	.	.	
1 messages	—	—	—	
2 documents	—	—	—	
2 unformatted	—	—	—	
2 data forms	—	—	—	
3 tabular	—	—	—	
4 graphic	—	—	—	
5 alarm/alert signals	—	—	—	
5.2. Source				
1 own display	—	—	—	
2 own files	—	—	—	
3 other users	—	—	—	
4 other files	—	—	—	
5 external	—	—	—	
6 [other]	—	—	—	_____
5.3. Destination				
1 own display	—	—	—	
2 own files	—	—	—	
3 local printer	—	—	—	
4 other users	—	—	—	
5 other files	—	—	—	
6 remote printer	—	—	—	
7 external	—	—	—	
8 [other]	—	—	—	_____

USI Capability	R	U	N
5.4. Transmission Control			
1. data specification	.	.	.
1 by source	—	—	—
2. by display name	.	.	.
1 all	—	—	—
2 designated part	—	—	—
3. by file name	.	.	.
1 all	—	—	—
2 designated part	—	—	—
4. by data name	.	.	.
1 category	—	—	—
2 item	—	—	—
2. routing	.	.	.
1. address headers	.	.	.
1 predefined	—	—	—
2 user-defined	—	—	—
2 subject descriptors	—	—	—
3. initiation	.	.	.
1. automatic	.	.	.
1 continuous	—	—	—
2 periodic	—	—	—
3 contingent	—	—	—
2 by request	—	—	—
5.5. Feedback			
1. data sent	.	.	.
1 initiated	—	—	—
2 confirmed	—	—	—
3 failed	—	—	—
2. data received	.	.	.
1 source	—	—	—
2 type	—	—	—
3 priority	—	—	—
3. availability	.	.	.
1. automatic	.	.	.
1 predefined	—	—	—
2 user-defined	—	—	—
2 by request	—	—	—
5.6. Queuing			
1 automatic	—	—	—
2 by request	—	—	—

USI Capability	R	U	N
5.7. Record Keeping			
1. log	.	.	.
1 automatic	—	—	—
2 by request	—	—	—
2. journal	.	.	.
1 automatic	—	—	—
2 by request	—	—	—
5.8. Design Change			
1. function	.	.	.
1 data type	—	—	—
2 source	—	—	—
3 destination	—	—	—
4 transmission control	—	—	—
5 feedback	—	—	—
6 queueing	—	—	—
7 record keeping	—	—	—
2. implementation	.	.	.
1. on-line	.	.	.
1 by transaction	—	—	—
2 by software change	—	—	—
2 off-line	—	—	—

USI Capability	R	U	N	
6.0 DATA PROTECTION				
6.1. User Identification				
1 user code	—	—	—	
2 station code	—	—	—	
3 job code	—	—	—	
4 project code	—	—	—	
5. password	.	.	.	
1. fixed	.	.	.	
1 assigned	—	—	—	
2 user-chosen	—	—	—	
2. changing	.	.	.	
1 assigned	—	—	—	
2 user-chosen	—	—	—	
6.2. Data Access				
1. user code	.	.	.	
1 for file	—	—	—	
2 for data category	—	—	—	
3 [other]	—	—	—	_____
2. password	.	.	.	
1 for file	—	—	—	
2 for data category	—	—	—	
3 [other]	—	—	—	_____
3. access record	.	.	.	
1 for user	—	—	—	
2 for file	—	—	—	
3 for data category	—	—	—	
4 [other]	—	—	—	_____
6.3. Data Change				
1. user code	.	.	.	
1 for file	—	—	—	
2 for data category	—	—	—	
3 [other]	—	—	—	_____
2. password	.	.	.	
1 for file	—	—	—	
2 for data category	—	—	—	
3 [other]	—	—	—	_____
3. change record	.	.	.	
1 for user	—	—	—	
2 for file	—	—	—	
3 for data category	—	—	—	
4 [other]	—	—	—	_____
4. error prevention	.	.	.	
1 data validation	—	—	—	
2 redundant entry	—	—	—	

USI Capability	R	U	N	
6.4. Data Transmission				
1. source	.	.	.	
1 automatic	—	—	—	
2 by request	—	—	—	
2. destination	.	.	.	
1 automatic	—	—	—	
2 by request	—	—	—	
3. transmission control	.	.	.	
1 automatic	—	—	—	
2 by request	—	—	—	
6.5. Loss Prevention				
1. reversible procedures	.	.	.	
1 BACKUP	—	—	—	
2 CONFIRM	—	—	—	
2. file protection	.	.	.	
1. SAVE	.	.	.	
1 automatic	—	—	—	
2 by request	—	—	—	
2. archive	.	.	.	
1 automatic	—	—	—	
2 by request	—	—	—	
3. data transmission	.	.	.	
1 parity check	—	—	—	
2 [other]	—	—	—	_____
6.6. Design Change				
1. type	.	.	.	
1 user identification	—	—	—	
2 data access	—	—	—	
3 data change	—	—	—	
4 data transmission	—	—	—	
5 loss prevention	—	—	—	
6 [other]	—	—	—	_____
2. implementation	.	.	.	
1. on-line	.	.	.	
1 by transaction	—	—	—	
2 by software change	—	—	—	
2 off-line	—	—	—	
3. change control	.	.	.	
1 data entry	—	—	—	
2 data display	—	—	—	
3 sequence control	—	—	—	
4 user guidance	—	—	—	
5 data transmission	—	—	—	
6 data protection	—	—	—	

APPENDIX B

DESIGN GUIDELINES FOR DATA ENTRY FUNCTIONS

Data entry refers to input by the user of data items to be processed. Command inputs or option selections intended to control data processing are considered separately in the discussion of sequence control (Appendix D). Data entry is heavily emphasized in tasks related to clerical jobs, and many other tasks involve data entry to some degree. Because data entry is so common, because the requirements of data entry seem to be readily understood, and because inefficiencies caused by poorly designed data entry are so apparent, many of the published recommendations for good USI design deal with this topic.

Design of data input transactions is necessarily influenced by hardware selection. For that reason, design guidelines for input devices receive considerable attention. A notable example is standardization of keyboard layouts. Future technological advances in input hardware may well influence the design of data entry tasks, presaged perhaps by the current advocacy of voice input. But the major need in information systems is for consistently good software design. It is in improving the logic of data entry that the chief gains can be made, and it is here that design guidance should prove most helpful.

Some ideas seem so basic that they are seldom expressed as explicit design principles. Here is an example: a user should not have to enter the same data twice. Now that is something every designer knows, even if it is sometimes forgotten. A corollary is this: a user should not have to enter a data item already entered by another user. That seems to be good common sense, although one could imagine occasional exceptions to the rule when cross validation of data inputs may be required.

How can duplicative data entry be avoided in practice? The solution lies in designing the USI (i.e., programming the computer) to maintain context. Thus when a user identifies a particular category of interest, for example a squadron of aircraft, the computer should be able to access all previously entered data relevant to that squadron and not require the user to enter such data again.

In repetitive data entry transactions the user should have some means of establishing context, for example by defining default entries for selected data items, in effect telling the computer those items will stay the same until the default value is changed or

removed. If the user enters one item of data about a particular squadron, it should be possible to enter a second item immediately thereafter without having to re-identify that squadron.

Context should also be preserved to help speed correction of input errors. One significant advantage of on-line data entry is the opportunity for immediate computer validation of user inputs, with timely feedback so that the user can correct detected errors while that set of entries is still fresh in his mind and/or while documented source data are still at hand. Here the computer should preserve the context of each data entry transaction, saving correct items so that the user does not have to enter those again while changing incorrect items.

Preservation of context is, of course, important in all aspects of user-system interaction, with implications for data display, sequence control and user guidance, as well as for data entry. The importance of context is emphasized again in the discussion of those other functional areas.

Another important design concept is that of flexibility. The idea that USI design should adapt flexibly to user needs is often expressed. The means of achieving such flexibility should be spelled out in USI guidelines. For data entry functions it is important that the pacing of inputs be controlled flexibly by the user. Tasks where the pacing of user inputs is set by a machine (for example, keying ZIP codes at an "automated" post office) are stressful and error-prone.

Aside from flexibility in pacing, the user will often benefit from having some flexible choice in the ordering of inputs. Although this kind of flexibility is related to the topic of sequence control, it merits discussion here as well. What is needed for USI design is some sort of suspense file(s) to permit flexible ordering of user inputs, including temporary omission of unknown items, backup to correct mistaken entries, cancellation of incomplete transactions, etc.

As noted earlier, the user may also benefit from flexibility in defining default options to simplify data entry during a sequence of transactions. Some systems include only those defaults anticipated by the designers, which may not prove helpful to the user in a particular instance.

These general concepts are represented in the specific design guidelines for data entry functions proposed in the following pages.

DATA ENTRY Objectives:

Minimized input actions by user
Low memory load on user
Consistency of data entry transactions
Compatibility of data entry with data display
Flexibility for user control of data entry

1.0 General

-1 When data entry is a significant task function, it should be accomplished via the user's primary display.

Example: Entry via typewriter is acceptable only if the typewriter itself, under computer control, is the primary display medium.

Comment: When the primary display is basically formatted for other purposes, such as a graphic display for process control, a separate "window" on the display may have to be reserved for data entry.

-2 Data entry transactions, and associated displays, should be designed so that the user can stay with one mode of entry as long as necessary for the data entry task, before having to shift to another.

Example: Shifts from lightpen to keyboard input and then back again should be minimized.

Comment: This, like other guidelines here, assumes a task-oriented user, busy or even overloaded, who needs efficiency of data entry.

Reference: EG 6.1.1.

-3 Keyed inputs should always appear in the display.

Exception: Passwords and other secure entries.

Reference: EG 6.3.7; MS 5.15.3.9.4.2.

-4 Keyed data entry on an electronic display should generally be accomplished by direct character replacement, in which keyed inputs replace underscores (or other delimiter symbols) in defined data fields.

Exception: For general text entry, no field delimiters are needed.

1.0 General (cont.)

-5 For data change, a consistent mode should be adopted, in which new entries replace previous entries (including default values, if any) either by direct character substitution, or by insertion and deletion.

Example: Text editing can be handled either way.

Comment: In many cases, direct modification of displayed data will reduce keying and permit more compact display formats. There may be some risk of user confusion, however, in replacement of an old value with a new one, during the transitional period when the item being changed is seen as a composite beginning with the new value and ending with the old.

Comment: In some cases it may help the user to key a new entry directly above or below display of the prior entry it will replace, if that is done consistently. Here the user can compare values before confirming entry of the new data and deletion of the old.

-6 Whenever possible, data entry should be self-paced, depending upon the user's needs, attention span and time available, rather than computer processing or external events.

Comment: When self-pacing does not seem feasible, the general approach to task allocation and USI design should be reconsidered.

-7 Data entry should not be slowed or paced by delays in control response; for normal operation, control delays or lockouts should not exceed 0.2 seconds.

Example: Key press followed by display of symbol.

Comment: This recommendation is intended to ensure efficient operation in routine, repetitive data entry tasks. Somewhat longer delays may be tolerable in special circumstances, perhaps to reduce variability in control response, or perhaps in cases where data entry comprises a relatively small portion of the user's task.

Reference: EG Table 2.

1.0 General (cont.)

-8 Data input should always require an explicit ENTER action, and not be accomplished as a side effect of some other action.

Example: Returning to a menu of control options should not by itself result in entry of data just keyed onto a display.

Comment: This practice permits the user to review data and correct errors before computer processing, particularly helpful when data entry is complex and/or difficult to reverse.

-9 An ENTER key should be explicitly labeled to indicate its function to the user.

Example: The ENTER key should not be labeled in terms of mechanism, such as CR or RETURN or XMIT.

Comment: For a computer-naive user, the label should perhaps be even more explicit, such as ENTER DATA. Ideally, one consistent ENTER label would be adopted for all systems and so become familiar to all users.

Reference: PR 3.3.9.

-10 When a stored data item is changed (or deleted) by direct command entry without first being displayed, then both the old and new values should be displayed so that the user can confirm or nullify the change before the transaction is completed.

Comment: This practice will tend to prevent inadvertent change, and is particularly useful in protecting delete actions. Like other recommendations intended to reduce error, it assumes that accuracy of data input is worth extra keying action by the user. For some tasks, that may not be true.

-11 Ideally, the length of an individual data entry should not exceed 5-7 characters.

Exception: Meaningful words and general textual material.

Comment: Longer items exceed the user's memory span, inducing errors in both data entry and data review.

Reference: BB 2.5.2; EG 6.3.3.

1.0 General (cont.)

-12 When a long data item must be entered, it should be partitioned into shorter symbol groups for both entry and display.

Example: A 10-digit telephone number can be entered as three groups, NNN-NNN-NNNN.

-13 When portions of a long item are highly familiar, ideally those should be entered last.

Exception: But not if that sequence would violate a functional requirement, such as initial keying of area code in telephone numbers, or if common usage puts the familiar first.

Comment: This practice will reduce the load on the user's short-term memory.

Reference: EG 6.3.4.

-14 Minimize data entry keying by abbreviating lengthy inputs, when that can be done without ambiguity.

Comment: Some flexibility should be provided for users of different ability. Novice and/or occasional users may prefer to make full-form entries, while experienced users will learn and benefit from appropriate abbreviations.

Reference: BB 6.4.1; EG 6.3.5.

-15 When abbreviations are used to shorten data entry, those abbreviations should follow some consistent rule that can be explained to the user.

Example: Simple truncation is probably the best choice.

Comment: It is important for both encoding and decoding abbreviations that the user know what the rule is. Truncation provides inexperienced users with a straightforward and highly successful method for generating abbreviations, and is a rule that can be easily explained. Moreover, truncation works at least as well as more complicated rules involving word contraction with omission of vowels, etc.

Reference: Moses and Ehrenreich, 1981.

1.0 General (cont.)

-16 Abbreviations should be of fixed length.

Comment: Desirable length will depend upon the vocabulary size of words to be abbreviated. For a vocabulary of 75 words, 4-letter abbreviations will suffice. For smaller vocabularies, shorter abbreviations can be used.

Reference: Moses and Ehrenreich, 1981.

-17 Special abbreviations (i.e., those not formed by consistent rule) should be used only when required for clarity.

Comment: Special abbreviations will be needed to distinguish between words whose abbreviations by rule are identical, or when abbreviation by rule forms another word, or when the special abbreviation is already familiar to system users. Such special cases should represent less than 10 percent of all abbreviations used.

Reference: Moses and Ehrenreich, 1981.

-18 When an abbreviation must deviate from the consistent rule, the extent of deviation should be minimized.

Example: Letters in the truncated form should be changed one at a time until a unique abbreviation is achieved.

Reference: Moses and Ehrenreich, 1981.

-19 When abbreviated data entries are not recognized, the computer should apply data validation routines and interrogate the user as necessary to resolve any ambiguity.

-20 When abbreviated codes are used to shorten data entry, code values should be designed to be meaningful and distinctive in order to avoid potentially confusing similarity.

Example: BOS vs. LAS is good; but LAX vs. LAS risks confusion.

1.0 General (cont.)

-21 When code values must be entered, menu selection should be considered as an appropriate dialogue mode, rather than keyed entry.

Comment: Menu selection will prove more accurate than keyed entry for arbitrary codes. Menu selection does not require the user to remember codes and does not require the user to key code entries accurately.

Reference: Siebel, 1972; Gade, Fields, Maisano and Marshall, 1980.

See also: Section 3.1.3.

-22 When alphabetic data entry is required, the user should be able to enter each letter with a single stroke of an appropriately labeled key.

Comment: More complex double-keying methods will require special user training, and will risk frequent data entry errors.

Comment: Software might be provided to interrogate the user to resolve any input ambiguities resulting from hardware limitations, such as when several letters are represented on each key of a button panel used primarily for numeric entry.

Reference: Smith and Goodwin, 1971a.

-23 Special characters requiring shift keying should be avoided insofar as possible.

Comment: Conversely, keyboard designers should put frequently used special characters where they can be easily keyed.

Reference: EG 6.3.12.

-24 Entry of leading zeros should be optional for general numeric input.

Exception: Special cases such as entry of serial numbers or other numeric identifiers.

Reference: BB 6.2.3; EG 6.3.11.

1.1 Position Designation (Cursor Control)

-1 Position designation on an electronic display should be accomplished by means of a movable cursor with distinctive visual features (shape, blink, etc.).

Exception: When position designation involves only selection among displayed alternatives, then some form of highlighting selected items might be used instead of a separately displayed cursor.

-2 If multiple cursors are used (e.g., one for alphanumeric entry, one for tracking, one for line drawing), they should be visually distinctive from one another.

-3 The cursor should be designed so that it does not obscure any other character displayed in the position designated by the cursor.

-4 When fine accuracy of positioning is required, as in some forms of graphic interaction, the displayed cursor should include a point designation feature.

Example: A cross may suffice (like cross-hairs in a telescope), or perhaps a notched or V-shaped symbol (like a gun sight).

-5 Actual entry ("activation") of a designated position should be accomplished by an explicit user action distinct from cursor placement.

Exception: Tracking tasks and other situations where the need for rapid input may override the need to reduce entry errors.

Reference: MS 5.15.3.9.2.4.b.

-6 Computer acceptance of a designated position should be signaled by direct feedback to the user within 0.2 seconds.

Example: Almost any consistently programmed display change will suffice, perhaps brightening or flashing a selected symbol; in some applications it may be desirable to provide an explicit message indicating that a selection has been made.

Reference: EG Table 2; MS 5.15.1.4.a, 5.15.3.9.2.4.a.

1.1 Position Designation (cont.)

-7 If there is a predefined HOME position for the cursor, which is usually the case, that position should be consistent on displays of a given type.

Example: HOME might be in the upper left corner of a text display, or at the first field in a form-filling display, or at the center of a graphic display.

Comment: The HOME position of the cursor should also be consistent in different windows/sections of a partitioned display.

-8 For arbitrary position designation, the cursor control should permit both fast movement and accurate placement.

Comment: Rough positioning should take no more than 0.5 seconds for a displacement of 20-30 cm on the display. Fine positioning may require incremental stepping of the cursor, or a control device incorporating a large control/display ratio for small displacements, or a selectable vernier mode of control use.

Reference: EG 6.1.

-9 The displayed cursor should be stable, i.e., should remain where it is placed until moved by the user (or computer) to another position.

Comment: Some special applications, such as aided tracking, may benefit from computer-controlled cursor movement. The intent of the recommendation here is to avoid unwanted "drift".

Reference: EG 6.1.

-10 When cursor positioning is incremental by discrete steps, the step size of cursor movement should be consistent in both right and left directions, and both up and down directions.

-11 When character size is variable on the display, incremental cursor positioning should have a variable step size corresponding to the size of currently selected characters.

1.1 Position Designation (cont.)

-12 If proportional spacing is used for displayed text, the computer should be programmed to make necessary adjustments automatically when the cursor is being positioned for data entry or data change.

Exception: Manual override may help the user in special cases where automatic spacing is not wanted.

Comment: The user cannot be relied upon to handle proportional spacing accurately.

-13 Continuous position designation, such as used for input of track data, should be accomplished by continuously operable controls (e.g., thumb wheel for one dimension, joystick for two dimensions) rather than by incremental, discrete key actions.

-14 When position designation is the sole or prime means of data entry, as in selection of displayed alternatives, cursor placement should be accomplished by a direct-pointing device (e.g., lightpen) rather than by incremental stepping or slewing controls (keys, joystick, etc.).

-15 In selection of displayed alternatives, the acceptable area for cursor placement should be made as large as consistently possible, including at least the area of the displayed label plus a half-character distance around the label.

Reference: EG 2.3.13, 6.1.3.

-16 When position designation is required in a task emphasizing keyed data entry, cursor movement should be controlled by some device integral to the keyboard (function keys, joystick, "cat", etc.) rather than by a separately manipulated device (lightpen, "mouse", etc.).

-17 Multiple cursors should be displayed only when justified by careful task analysis.

Comment: Multiple cursors may confuse a user, and so require special consideration if advocated in USI design.

-18 If multiple cursors are controlled by a single device, then a clear signal must be provided to indicate to the user which cursor is currently under control.

1.1 Position Designation (cont.)

-19 If multiple cursors are controlled by different devices, their separate controls should be compatible in operation.

Reference: Morrill and Davies, 1961.

-20 On initial appearance of a form-filling data entry display, the cursor should be placed automatically at the first character position of the first input field.

-21 Displays for form-filling data input should be designed so as to minimize user actions required for cursor movement from one entry field to the next.

-22 Sequential cursor positioning in predefined areas, such as displayed data entry fields, should be accomplished by programmable tab keys.

Comment: Automatic cursor advance is generally not desirable.

See also: 1.4-13.

-23 Areas of a display not needed for data entry (such as labels and blank spaces) should be made inaccessible to the user, under computer control, so that the cursor does not have to be stepped through blank areas nor are they sensitive to pointing actions.

Exception: When it is expected that a user may have to modify display formats, such automatic format protection can be handled as a general default option subject to user override.

Comment: Mechanical overlays on the display should not be used for format protection.

Reference: EG 7.5; PR 3.3.2.

-24 User action confirming entry of multiple data items should result in input of all items, regardless of where the cursor is placed on the display.

1.2 Direction Designation

-1 When designation of direction (azimuth, bearing, heading, etc.) is based on already quantified data, then keyed entry should be used.

-2 When direction designation is based on graphic representation, then some "analog" means of entry should be provided, such as vector rotation on the display, and/or a suitably designed rotary switch.

Example: Heading estimation for displayed radar trails.

Exception: When approximate direction designation will suffice, for just eight cardinal points, keyed entry can be used.

Comment: In matching graphic display, an entry device providing a visual analog will prove both faster and more accurate.

Reference: Smith, 1962a.

1.3 Text

(guidelines deferred pending further information)

1.4 Data Forms

-1 Using a form-filling dialogue, entry of logically related items should be accomplished by a single, explicit action at the end, rather than by separate entry of each item.

Comment: This practice permits user review and possible data correction prior to entry, and also clarifies for the user just when grouped data are processed. It will also permit efficient cross validation of related data items by the computer.

Comment: Depending on form design, this practice might involve entering the entire form, or entry by page or section of a longer form. Just where entry is required should be indicated to the user.

See also: 1.0-8.

1.4 Data Forms (cont.)

-2 When multiple data items are entered as a single transaction, as in form filling, the user should be allowed to RESTART, CANCEL, or BACKUP and change any item before taking a final ENTER action.

Reference: BB 6.9; MS 5.15.1.2.4, 5.15.3.9.4.1.

See also: 3.5-2

-3 Whenever possible, multiple data items should be entered without the need for special separators or delimiters, either by keying into predefined entry fields or by including simple spaces between sequentially keyed items.

-4 When a field delimiter must be used for data entry, a standard character should be adopted for that purpose; slash (/) is recommended.

-5 For all dialogue types involving prompting, data entries should be prompted explicitly by means of displayed labels for data entry fields, and/or associated user guidance messages.

-6 Field labels should consistently indicate what data items are to be entered.

Example: A field labeled NAME should always require name entry, and not sometimes require something different like elevation.

-7 In ordinary use, field labels should be protected and transparent to keyboard control, so that the cursor skips over them when spacing or tabbing.

Reference: PR 3.3.2, 4.8.1.

-8 Special characters should be used to delineate each entry field; an underscore is recommended.

Comment: Such implicit prompts help reduce data entry errors by the user.

Reference: BB 6.2.1; EG 6.3, 6.3.1; PR 4.8.1.

1.4 Data Forms (cont.)

-9 Field delineation cues should indicate a fixed or maximum acceptable length of the entry.

Comment: This method of prompting is more effective than simply telling the user how long an entry should be, e.g., "Enter ID (12 digits)". Underscoring gives a direct visual cue as to the number of characters to be entered, and the user does not have to count them.

Reference: BB 6.2.1; EG 6.3; PR 4.8.2.

-10 Similar implicit cues should be provided when data entry is prompted by auditory displays.

Example: Tone codes can be used to indicate the type and length of requested data entries.

Reference: Smith and Goodwin, 1970.

-11 Field delineation cues should distinguish required from optional entries.

Example: A solid underscore might be used to indicate required entries, a dotted underscore optional entries.

Reference: BB 6.6; PR 4.8.6.

-12 When item length is variable, the user should not have to justify an entry either right or left, and should not have to remove any unused underscores; computer processing should handle those details automatically.

Reference: BB 6.2.2; EG 6.3.2.

-13 When multiple items (especially those of variable length) will be entered by a skilled touch typist, each entry field should end with an extra (blank) character space; software should be designed to prevent keying into a blank space, and an auditory signal should be provided to alert the user when that happens.

Comment: This will permit consistent use of tab keying to move from one field to the next.

Reference: PR 4.9.1.

See also: 1.1-22.

1.4 Data Forms (cont.)

-14 When entry fields are distributed across a display, a consistent format should be adopted for relating labels to delineated entry areas.

Example: The label might always be to the left of the field; or the label might always be immediately above and left-justified with the beginning of the field.

Comment: Such consistent practice will help the user distinguish labels from data in distributed displays.

-15 Labels for data entry fields should be distinctively worded, so that they will not be readily confused with data entries, labeled control options, guidance messages, or other displayed material.

-16 In labeling data entry fields, only agreed terms, codes and/or abbreviations should be used.

Comment: Do not create new jargon; if in doubt, pretest all proposed wording with a sample of qualified users.

Reference: BB 6.1.5; PR 4.5.6.

See also: 2.1.1-20.

-17 The label for each entry field should end with a special symbol, signifying that an entry may be made.

Example: A colon is recommended for this purpose, e.g., NAME: _____

Comment: A symbol should be chosen that can be reserved exclusively for prompting user inputs, or else is rarely used for any other purpose.

Reference: BB 6.5.

-18 Labels for entry fields may incorporate additional cueing of data formats when that seems helpful.

Example: DATE (MDY): __/__/__

Example: DATE: __/__/__
MM DD YY

Reference: PR 4.8.9.

1.4 Data Forms (cont.)

-19 When a dimensional unit is consistently associated with a particular data field, it should be displayed as part of the fixed label rather than entered by the user.

Example: COST: $ _____

Example: SPEED (MPH): _____

Reference: PR 4.8.11.

-20 When alternative dimensional descriptors are acceptable, then space should be provided in the data field for user entry of a unit designator.

Example: DISTANCE: _____ MI/KM: __

Reference: PR 4.8.11.

-21 Data should be entered in units that are familiar to the user.

Comment: Data conversion, if necessary, should be handled by the computer.

Reference: BB 6.3.

-22 When data entry displays are crowded, auxiliary coding should be adopted to distinguish labels from data.

Example: A recommended practice is to display fixed, familiar labels in dim characters, with data entries bright.

Comment: For novice users, it may sometimes be helpful to have brighter labels, if that could be provided as a selectable option.

Reference: PR 3.3.2.

See also: 2.1.2-7.

1.4 Data Forms (cont.)

-23 The display format for data entry should be compatible with whatever format is used for display output, scanning and review of the same data; item labels and ordering should be preserved consistently from one display to the other.

Comment: When a display format optimized for data entry seems unsuited for data display, or vice versa, some compromise format should be designed taking into account the relative functional importance of data entry and data review in the user's task.

See also: 2.3-2.

-24 When data entry involves transcription from source documents, form-filling displays should match (or be compatible with) paper forms; in a question-and-answer dialogue, the sequence of entry should match the data sequence in source documents.

Comment: When paper forms are not optimal for data entry, consider revising the layout of the paper form.

Comment: When data entries must follow an arbitrary sequence of external information (e.g., keying telephoned reservation data), some form of command language dialogue should be used instead of form filling, to identify each item as it is entered so that the user does not have to remember and re-order items.

Reference: BB 2.8.9; PR 4.8.3, 4.8.5.

See also: 2.3-2.

-25 If no source document or external information is involved, the ordering of multiple-item data entries should follow the logical sequence in which the user can be expected to think of them.

Comment: Alternatively, data entry can sometimes be made more efficient by placing all required fields before any optional fields.

Reference: BB 6.6; PR 4.8.5.

See also: 2.3-2.

1.4 Data Forms (cont.)

-26 When a form for data input is displayed, the cursor should be positioned automatically in the first entry field.

Exception: If a data form is regenerated following an entry error, the cursor should be positioned in the first field in which an error has been detected.

Reference: PR 4.9.1.

1.5 Tabular Data

-1 When sets of data items must be entered sequentially, in a repetitive series, a tabular format where data sets are keyed row by row should be considered to facilitate the process.

Exception: When the items in each data set will exceed the display capacity of a single row, tabular entry will usually not be desirable.

Reference: PR 4.8.4.

-2 Column headers and row labels should be worded informatively, so as to help guide data entry.

-3 Column headers and row labels should be formatted distinctively, so as to distinguish them from data entries.

-4 When tabular formats are used for data entry, column labels should be left-justified with the leftmost position beginning column entries.

Comment: This consistent practice will prove especially helpful when columns vary in width.

-5 Justification of tabular data entries should be handled automatically by the computer; the user should not have to enter any leading blanks or other extraneous formatting characters.

Reference: BB 6.2.3.

1.5 Tabular Data (cont.)

-6 It should be possible for the user to make numeric entries (e.g., dollars and cents) as left-justified, but they should be automatically justified with respect to a fixed decimal point when a display of those data is subsequently regenerated for review by the user.

Reference: PR 4.8.10.

-7 For input of tabular data, when vertical repetition of entries is frequent the user should be provided a DITTO key to speed entry of duplicative data.

-8 For dense tables, those with many row entries, some extra visual cue should be provided to guide the user accurately across columns.

Example: A blank line after every fifth row is recommended. Alternatively, adding dots between columns at every fifth row may suffice.

Comment: This practice is probably more critical for accurate data review and change than it is for initial data entry, but is desirable in the interest of compatible display formats.

1.6 Graphic Data

(no guidelines presently available)

1.7 Data Validation

-1 Automatic data validation software should be incorporated to check any entry whose input and/or correct format or content is required for subsequent data processing.

Comment: Do not rely on the user always to make correct inputs. When validity of data entries can be established automatically, computer checking will be more accurate.

Reference: MS 5.15.1.2.2; PR 4.12.4.

1.4 Data Forms (cont.)

-26 When a form for data input is displayed, the cursor should be positioned automatically in the first entry field.

Exception: If a data form is regenerated following an entry error, the cursor should be positioned in the first field in which an error has been detected.

Reference: PR 4.9.1.

1.5 Tabular Data

-1 When sets of data items must be entered sequentially, in a repetitive series, a tabular format where data sets are keyed row by row should be considered to facilitate the process.

Exception: When the items in each data set will exceed the display capacity of a single row, tabular entry will usually not be desirable.

Reference: PR 4.8.4.

-2 Column headers and row labels should be worded informatively, so as to help guide data entry.

-3 Column headers and row labels should be formatted distinctively, so as to distinguish them from data entries.

-4 When tabular formats are used for data entry, column labels should be left-justified with the leftmost position beginning column entries.

Comment: This consistent practice will prove especially helpful when columns vary in width.

-5 Justification of tabular data entries should be handled automatically by the computer; the user should not have to enter any leading blanks or other extraneous formatting characters.

Reference: BB 6.2.3.

1.5 Tabular Data (cont.)

-6 It should be possible for the user to make numeric entries (e.g., dollars and cents) as left-justified, but they should be automatically justified with respect to a fixed decimal point when a display of those data is subsequently regenerated for review by the user.

Reference: PR 4.8.10.

-7 For input of tabular data, when vertical repetition of entries is frequent the user should be provided a DITTO key to speed entry of duplicative data.

-8 For dense tables, those with many row entries, some extra visual cue should be provided to guide the user accurately across columns.

Example: A blank line after every fifth row is recommended. Alternatively, adding dots between columns at every fifth row may suffice.

Comment: This practice is probably more critical for accurate data review and change than it is for initial data entry, but is desirable in the interest of compatible display formats.

1.6 Graphic Data

(no guidelines presently available)

1.7 Data Validation

-1 Automatic data validation software should be incorporated to check any entry whose input and/or correct format or content is required for subsequent data processing.

Comment: Do not rely on the user always to make correct inputs. When validity of data entries can be established automatically, computer checking will be more accurate.

Reference: MS 5.15.1.2.2; PR 4.12.4.

1.4 Data Forms (cont.)

-26 When a form for data input is displayed, the cursor should be positioned automatically in the first entry field.

Exception: If a data form is regenerated following an entry error, the cursor should be positioned in the first field in which an error has been detected.

Reference: PR 4.9.1.

1.5 Tabular Data

-1 When sets of data items must be entered sequentially, in a repetitive series, a tabular format where data sets are keyed row by row should be considered to facilitate the process.

Exception: When the items in each data set will exceed the display capacity of a single row, tabular entry will usually not be desirable.

Reference: PR 4.8.4.

-2 Column headers and row labels should be worded informatively, so as to help guide data entry.

-3 Column headers and row labels should be formatted distinctively, so as to distinguish them from data entries.

-4 When tabular formats are used for data entry, column labels should be left-justified with the leftmost position beginning column entries.

Comment: This consistent practice will prove especially helpful when columns vary in width.

-5 Justification of tabular data entries should be handled automatically by the computer; the user should not have to enter any leading blanks or other extraneous formatting characters.

Reference: BB 6.2.3.

1.5 Tabular Data (cont.)

-6 It should be possible for the user to make numeric entries (e.g., dollars and cents) as left-justified, but they should be automatically justified with respect to a fixed decimal point when a display of those data is subsequently regenerated for review by the user.

Reference: PR 4.8.10.

-7 For input of tabular data, when vertical repetition of entries is frequent the user should be provided a DITTO key to speed entry of duplicative data.

-8 For dense tables, those with many row entries, some extra visual cue should be provided to guide the user accurately across columns.

Example: A blank line after every fifth row is recommended. Alternatively, adding dots between columns at every fifth row may suffice.

Comment: This practice is probably more critical for accurate data review and change than it is for initial data entry, but is desirable in the interest of compatible display formats.

1.6 Graphic Data

(no guidelines presently available)

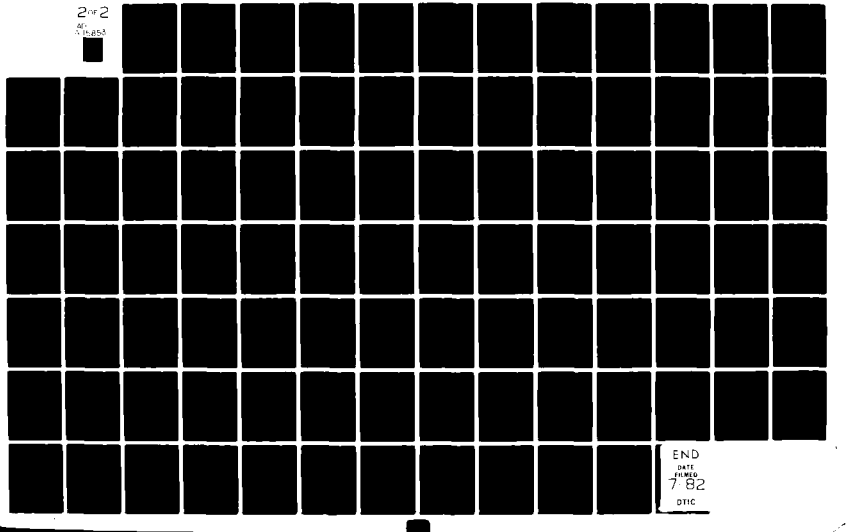
1.7 Data Validation

-1 Automatic data validation software should be incorporated to check any entry whose input and/or correct format or content is required for subsequent data processing.

Comment: Do not rely on the user always to make correct inputs. When validity of data entries can be established automatically, computer checking will be more accurate.

Reference: MS 5.15.1.2.2; PR 4.12.4.

AD-A115 853 MITRE CORP BEDFORD MA F/G 9/2
USER-SYSTEM INTERFACE DESIGN FOR COMPUTER-BASED INFORMATION SYS--ETC(U)
APR 82 S L SMITH F19628-81-C-0001
UNCLASSIFIED MTR-8464 ESD-TR-82-132 NL



1.7 Data Validation (cont.)

-2 When required data entries have not been input, but can be deferred, data validation software should signal that omission to the user, permitting either immediate or delayed input of missing items.

Reference: PR 4.8.7.

-3 When entry of a required data item is deferred, the user should have to enter a special symbol in the data field to indicate that the item has been temporarily omitted rather than ignored.

Reference: PR 4.8.7, 4.12.2.

-4 In a repetitive data entry task, data validation for one transaction should be completed, and the user allowed to correct errors, before another transaction can begin.

Comment: This is particularly important when the user is transcribing data from source documents, so that detected input errors can be corrected while the relevant document is still at hand.

-5 If item-by-item data validation within a multiple-entry transaction is provided, it should only be as a selectable option.

Comment: This capability will sometimes help a novice user, who may be uncertain about what requirements are imposed on each data item; but it may slow a skilled user if the computer processing delays next item entry.

Reference: EG 6.3.9.

-6 When helpful default values for data entry cannot be predicted by USI designers, which is often the case, the user (or perhaps some authorized supervisor) should have a special transaction to define, change or remove default values for any data entry field.

1.7 Data Validation (cont.)

-7 On initiation of a data entry transaction, currently defined default values should be displayed automatically in their appropriate data fields.

Comment: The user should not be expected to remember them.

Comment: It may be helpful to mark or highlight default values in some way to distinguish them from new data entries.

-8 User acceptance of a displayed default value for entry should be accomplished by simple means, such as by a single confirming key input, or simply by tabbing past the default field.

Comment: Similar techniques should be used in tasks involving user review of previously entered data.

-9 In any data input transaction the user should be able to replace a default value with a different entry, without necessarily changing the default definition for subsequent transactions.

1.8 Other Data Processing

-1 A user should not be required to enter "bookkeeping" data that the computer could determine automatically.

Example: A user generally should not have to identify his work station to initiate a transaction, nor include other routine data such as transaction sequence codes.

Comment: Complicated data entry routines imposed in the interest of security may hinder the user in achieving effective task performance; other means of ensuring data security should be considered.

1.8 Other Data Processing (cont.)

-2 A user should not be required to enter redundant data already accessible to the computer.

Example: The user should not have to enter both an item name and identification code when either one defines the other.

Exception: As needed for resolving ambiguous entries, for user training, or for security (e.g., user identification).

Comment: Verification of previously entered data is often better handled by review and confirmation rather than by re-entry.

Reference: EG 6.3.10.

-3 Data entries made in one transaction should be retrieved by the computer when relevant to another transaction, and displayed for user review if appropriate.

Comment: The user should not have to enter such data again.

Reference: BB 6.4.2.

-4 Whenever needed, automatic computation of derived data should be provided, so that a user does not have to calculate and enter any number that can be derived from data already accessible to the computer.

-5 Whenever needed, automatic cross-file updating should be provided, so that a user does not have to enter the same data twice.

Example: Assignment of aircraft to a mission should automatically indicate that commitment in squadron status files as well as in a mission assignment file.

1.9 Design Change

(no guidelines presently available)

APPENDIX C

DESIGN GUIDELINES FOR DATA DISPLAY FUNCTIONS

Data display, i.e., some kind of output from a computer to its users, is needed for all data handling tasks. Data display is emphasized particularly in monitoring and control tasks. Included as data display may be hardcopy printouts as well as more mutable electronic displays. Also included are auxiliary displays and signaling devices, including voice output, which may alert the user to unusual conditions. Displays specifically intended to guide the user in his interaction with the system are discussed separately under the topic of sequence control (Appendix D).

In general, it may be said that rather less is known about data display, and information assimilation by the user, than about data entry. In current information system design, display formatting is an art. Guidelines are surely needed.

Here again some general concepts deserve emphasis, including the importance of context and flexibility. Data displays must always be interpreted in the context of task requirements and user expectations. An early statement of the need for relevance in data display seems valid still:

> When we examine the process of man-computer communication from the human point of view, it is useful to make explicit a distinction which might be described as contrasting "information" with "data." Used in this sense, information can be regarded as the answer to a question, whereas data are the raw materials from which information is extracted. A man's questions may be vague, such as, "What's going on here?" or "What should I do now?" Or they may be much more specific. But if the data presented to him are not relevant to some explicit or implicit question, they will be meaningless. . . .
>
> What the computer can actually provide the man are displays of data. What information he is able to extract from those displays is indicated by his responses. How effectively the data are processed, organized, and arranged prior to presentation will determine how effectively he can and will extract the information he requires from his display. Too frequently these two terms <u>data</u> and <u>information</u> are confused, and the statement, "I need more information," is assumed to mean, "I want more symbols." The reason for the statement, usually, is that

the required information is not being extracted from the data. Unless the confusion between data and information is removed, attempts to increase information in a display are directed at obtaining more data, and the trouble is exaggerated rather than relieved.

(Smith, 1963b, pages 296-297)

Certainly this distinction between data and information should be familiar to psychologists, who must customarily distinguish between a physical stimulus (e.g., "intensity" of a light) and its perceived effect ("brightness"). The distinction is not familiar to system designers, however, although the issue itself is often addressed. In the following description of what has been called the "information explosion", notice how the terms data and information are used interchangeably, confounding an otherwise incisive (and lively) analysis:

> The sum total of human knowledge changed very slowly prior to the relatively recent beginnings of scientific thought. But it has been estimated that by 1800 it was doubling every 50 years; by 1950, doubling every 10 years; and by 1970, doubling every 5 years. . . . This is a much greater growth rate than an exponential increase. In many fields, even one as old as medicine, more reports have been written in the last 20 years than in all prior human history. And now the use of the computer vastly multiplies the rate at which information can be generated. The weight of the drawings of a jet plane is greater than the weight of the plane. The computer files of current IBM customer orders contain more than 100 billion bits of information -- more than the information in a library of 50,000 books.
>
> For man, this is a hostile environment. His mind could no more cope with this deluge of data, than his body could cope with outer space. He needs protection. The computer -- in part the cause of the problem -- is also the solution to the problem. The computer will insulate man from the raging torrents of information that are descending upon him.
>
> The information of the computerized society will be gathered, indexed, and stored in vast data banks by the computers. When man needs a small item of information he will request it from the computers. The machines, to satisfy his need, will sometimes carry on a simple dialogue with him until he obtains the data he wants.

> With the early computers, a manager would often have dumped on his desk an indigestible printout -- sometimes several hundred pages long. Now the manager is more likely to request information when he needs it, and receive data about a single item or situation on a screen or small printer.
>
> It is as though man were surviving in the depths of this sea of information in a bathyscaphe. Life in the bathyscaphe is simple, protected as it is from the pressure of the vast quantities of data. Every now and then man peers through the windows of the bathyscaphe to obtain facts that are necessary for some purpose or other. The facts that he obtains at any one time are no more than his animal brain can handle. The information windows must be designed so that man, with his limited capabilities, can locate the data he wants and obtain simple answers to questions that may need complex processing.
>
> (Martin, 1973, page 6)

Somehow a means must be found to provide and maintain context in data displays so that the user can find the information he needs for his job. Task analysis may point the way here, indicating what data are relevant to each stage of task performance. Design guidelines must emphasize the value of displaying no more data than the user needs, maintaining consistent display formats so that the user always knows where to look for different kinds of information, and using consistent labeling to help the user relate different data items, on any one display and from one display to another.

Detailed user information requirements will vary from time to time, however, and may not be completely predictable in advance, even from a careful task analysis. Here is where flexibility is needed, so that data displays can be tailored on-line to user needs. Such flexibility is sometimes provided through optional category selection, display offset and expansion features. If such options for display coverage are available, the user may be able to adjust his processing of data outputs in a way analogous to self pacing of data inputs.

In tasks where a user must both enter and retrieve data, which is often the case, the formatting of data displays should be compatible with the methods used for data entry. As an example, if data entry is accomplished via a form-filling dialogue, with a particular format for data fields, subsequent retrieval of that data set should produce an output display with the same format, especially if the user is expected to make changes to displayed

data, and/or additional entries. Where compaction of data output is required for greater efficiency, to review multiple data sets in a single display frame, the displayed items should retain at least an ordering and labeling compatible with those fields used previously for data entry.

Display design should also be compatible with dialogue types used for sequence control, and with hardware capabilities. Where user inputs are made via menu selection, using a pointing device like a lightpen, then display formats should give prominence (and adequate separation) to the labeled, lightpennable options. Location of multi-function keys at the display margin, to be labeled on the adjacent portion of the display itself, may provide flexibility for both data entry and sequence control, but will necessarily constrain the formatting of displays for data output.

These general concepts underlie many of the specific guidelines for data display that are proposed in the following pages.

DATA DISPLAY Objectives:

Efficient information assimilation by the user
Low memory load on user
Consistency of display format
Compatibility of data display with data entry
Flexibility for user control of data display

2.0 General

-1 Displayed data should be tailored to user needs, providing only necessary and immediately usable information at any step in a transaction sequence.

Comment: When user needs cannot be exactly anticipated by the designer, provision should be made for on-line tailoring of displays by the user, including data selection, display coverage and display suppression.

Reference: EG 3.1.4, 3.3.1; MS 5.15.2.3.

-2 Data should be displayed to the user in directly usable form.

Example: Too cryptic: "Error 459 in column 64." This is better: "Error 459, character in NAME entry cannot be recognized."

Comment: The user should not be required to transpose, compute, interpolate, translate displayed data into other units, or refer to documentation to determine the meaning of displayed data.

Reference: BB 3.3; EG 3.3.4; MS 5.15.2.8, 5.15.4.9.

-3 Data should be displayed consistently, following standards and conventions familiar to the user.

Example: If users work with metric units of measurement, do not display data in English units, or vice versa.

Example: Computer time records that are not in directly usable format should be converted for display, to a conventional 12-hour (AM/PM) clock or a 24-hour clock, in local time or whatever other time standard is appropriate to user needs.

Comment: Adopt a consistent standard when no specific user convention has been established.

Reference: BB 3.4; EG 2.2.4.

2.0 General (cont.)

-4 An experienced user should be provided means to control the amount, format, and complexity of displayed data.

Reference: EG 3.4.2.

-5 Data displays should permit direct user change (replacement) of displayed items, or entry of new items, when that represents an efficient transaction sequence.

Comment: Some consistent formatting cue, perhaps different initial cursor placement, should be provided to inform the user when displayed data can or cannot be changed.

Reference: PR 4.4.

-6 When integrity of displayed data is essential, maintain computer control over the display and do not permit the user to change controlled items.

Comment: Never assume voluntary compliance with instructions by the user, who may attempt unwanted changes by mistake, or for curiosity, or to subvert the system.

Reference: EG 3.4.8.

-7 If data must be remembered from one display to another, no more than 4-6 items should be displayed.

Comment: Better yet, do not require the user to rely on memory, but recapitulate needed items on the succeeding display.

Reference: EG 2.3.14, 2.3.15.

2.1 Data Type

2.1.1 Text

-1 When textual material is formatted, as in structured messages, headers, etc., consistent format should be maintained from one display to another.

Comment: A missing item should be shown as a labeled blank, rather than being omitted from a standard format.

-2 Running text (prose) should be displayed conventionally, in mixed upper and lower case.

Exception: An item intended to attract the user's attention, such as a label or title, may be displayed in upper case.

Exception: Upper case should be used when lower case letters will have decreased legibility, which is true on display terminals that cannot show descenders for lower case letters.

Comment: Normal reading of text is easier with conventional use of capitalization, i.e., to start sentences, indicate proper nouns and acronyms, etc.

Reference: BB 2.6; EG 3.4.3.

-3 Displayed text should be left justified to maintain constant spacing between words.

Exception: When right justification can be achieved by variable spacing, maintaining constant proportional differences in spacing between and within words, and consistent spacing between words in a line.

Comment: Reading is easier with constant spacing, which outweighs the debatable esthetic advantage of an even right margin achieved at the cost of uneven spacing. Uneven spacing is a greater problem with narrow column formats than with wide columns. Uneven spacing handicaps poor readers more than good readers.

Reference: PR 4.5.1, 4.10.5; Gregory and Poulton, 1970; Campbell, Marchetti and Mewhort, 1981.

2.1.1 Text (cont.)

-4 In textual material, words should be displayed intact wherever possible, with minimal breaking by hyphenation between lines.

Comment: Text is more readable if the entire word is on one line, even if that makes the right margin more ragged.

Reference: BB 3.2; EG 2.2.10; MS 5.15.4.9.g.

-5 Displayed paragraphs of text should be separated by at least one blank line.

Reference: EG 2.3.4.

-6 In textual display, every sentence should end with a period.

Reference: EG 2.2.13.

-7 In textual display, the main topic of each sentence should be placed near the beginning of the sentence.

Reference: BB 3.8.2.

-8 In textual display, short, simple sentences should be used.

Comment: Long sentences with multiple clauses may confuse the user. Consider breaking long sentences into two or more shorter statements.

Reference: BB 3.8, 3.8.1; EG 2.2.12; Wright and Reid, 1973.

-9 When speed of display output for textual material is slower than the user's normal reading speed, an extra effort should be made to word the text concisely.

Comment: Assume a normal average reading speed of 250 words per minute.

Reference: EG 3.3.7.

2.1.1 Text (cont.)

-10 In textual display, affirmative statements should be used rather than negative statements.

Example: This is potentially confusing: "Do not enter data before clearing the screen." This is better: "Clear the screen before entering data."

Comment: Tell the user what to do, rather than what to avoid.

Reference: BB 3.8.3.

-11 In textual display, sentences in the active voice should be used rather than passive voice.

Example: "The screen is cleared by pressing RESET" is too indirect. "Clear the screen by pressing RESET" is better.

Comment: Active voice sentences are generally easier to understand.

Reference: BB 3.8.5.

-12 In textual display, sentences describing a sequence of events should be phrased with a corresponding word order.

Example: Reverse order may confuse the user: "Before running programs, enter LOGON." Temporal order is clearer: Enter LOGON before running programs."

Reference: BB 3.8.6.

-13 When listing textual material, or other data, each item should start on a new line, i.e., the list should be a single column.

Exception: Multiple columns of data should be used where that facilitates comparison of corresponding data sets, as in tabular displays.

Exception: Multiple columns of data may be considered where shortage of display space dictates a compact format.

Reference: BB 2.3.2, 2.9.2; EG 2.3.5.

2.1.1 Text (cont.)

-14 Lists within text should be ordered by some logical principle; long lists, with more than seven items, should be ordered alphabetically.

Reference: EG 2.3.1.

See also: 2.3-6.

-15 If a list is displayed in multiple columns, item ordering should be vertical within each column, considered sequentially from left to right.

-16 For material that must be remembered in a displayed statement or list, the hardest items should be put at the beginning and the easiest items in the middle.

Reference: EG 3.3.3, 3.3.5.

-17 Material that need be recalled only for the next immediate user action should be put at the end of a statement or list.

Reference: EG 3.3.5.

-18 Text displays and labels should be worded from the viewpoint of the user, and not that of the system designer or programmer.

Reference: BB 2.2.2, 3.7.2; EG 3.4.5, 4.2.13.

-19 In text displays and labels, word usage should be consistent, particularly for technical terms.

Comment: Standard terminology should be defined and documented for reference by interface designers and by users.

Reference: BB 2.2.2; EG 3.4.5, 4.2.13; MS 5.15.1.3.1.

-20 In text displays and labels, word usage should avoid jargon terms.

Comment: When in doubt, pretest the meaning of words for prospective users, to ensure that there is no ambiguity.

Reference: BB 2.2.2, 3.7.1, 3.7.4; PR 4.5.6.

See also: 1.4-16.

2.1.1 Text (cont.)

-21 Wording of text and labels should be consistent from one display to another.

Example: The title of a display should be identical to the menu option used to request the display.

Reference: BB 3.7.3.

-22 In text display and labels, use distinct words rather than contractions or combined forms, especially in phrases involving negation.

Example: Use "will not" rather than "won't", "not complete" rather than "incomplete".

Comment: This practice will help the user understand the sense of the message.

Reference: BB 3.1.4; EG 2.2.15.

-23 In text display and labels, complete words should be used in preference to abbreviations.

Exception: Abbreviations may be displayed if they are significantly shorter, save needed space, and will be understood by the prospective users.

Exception: When abbreviations are required (or useful) for data entry, then corresponding use of those abbreviations in data display may help a user learn them for data entry.

Reference: BB 3.1.1; EG 4.1.3; MS 5.15.1.3.2.

-24 In text display and labels, when words are abbreviated, the designer should ensure that abbreviations are consistent in form, and that abbreviations of different words are distinguishable.

Reference: BB 3.1, 3.1.2; EG 4.1.3; MS 5.15.1.3.3; PR 4.5.6.

See also: 1.0-14 thru 1.0-18.

-25 If an abbreviation deviates from the consistent form, it should be specially marked whenever it is displayed.

Reference: Moses and Ehrenreich, 1981.

2.1.1 Text (cont.)

-26 When abbreviations are used, a dictionary of abbreviations should be available for on-line user reference.

Reference: BB 3.1.3.

-27 In text display and labels, abbreviations and acronyms should not include punctuation.

Example: Use "USAF" instead of "U.S.A.F."

Exception: Punctuation should be retained when needed for clarity, e.g., "4-in. front dimension" rather than "4 in front dimension".

Reference: BB 2.3.4; EG 2.2.14.

2.1.2 Data Forms

-1 When data items are displayed in a distributed format, each field should have an associated label to identify it.

Comment: Do not assume that the user can identify individual data fields because of past familiarity. Context may play a significant role: 617-271-4768 might be recognized as a telephone number if seen in a telephone directory, but might not be recognized as such in an unlabeled display.

Reference: BB 2.8.7; EG 2.2.16; MS 5.15.4.9.i.

-2 The field label should be a descriptive title, phrase or word, positioned adjacent to (above, or to the left of) a displayed item, or group of items.

Comment: Labels should be worded carefully to assist a new user in scanning the display and assimilating information quickly.

Comment: Labels may be worded as a heading or title reflecting the question for which the user seeks an answer in the data that follow.

Reference: BB 2.9.1; EG 3.2, 3.2.4; MS 5.15.2.10.

2.1.2 Data Forms (cont.)

-3 Field labels should be distinctive from one another in wording, to aid user discrimination.

Reference: BB 3.5; EG 3.2.3; MS 5.15.2.10.c.

-4 Consistent grammatical construction should be used for labels, and for items in a list.

Example: Do not use single words or phrases for some items and short sentences for others.

Reference: BB 3.8.4.

-5 Labels should be distinctive in format/positioning to help distinguish them from displayed data and other types of displayed material (e.g., error messages).

Reference: EG 3.2.3; MS 5.15.2.10.a.

-6 Labels and their associated data fields should be separated by at least one space in the display.

Reference: BB 2.9.5; EG 2.3.8.

-7 An option should be provided to highlight labels for a new user, or to dim labels for an experienced user.

Reference: EG 3.2; MS 5.15.2.10.b.

See also: 1.4-22.

-8 In data forms, labels and data fields should be consistently formatted, and aligned to minimize search time by the user.

Example: In a numbered list, vertically formatted, the data items should start in a fixed column position on the display.

Reference: EG 2.3.7, 2.3.9.

-9 The units of measurement for displayed data should be included either in the label or as part of each data item.

Reference: BB 2.8.8.

2.1.2 Data Forms (cont.)

-10 The ordering and layout of corresponding data fields should be consistent from one display to another.

Reference: BB 2.8.3.

-11 The detailed internal format of frequently used data fields should be consistent from one display to another.

Example: Telephone numbers should be consistently hyphenated, as 213-394-1811.

Example: Time records might be consistently formatted with colons, as HH:MM:SS, or HH:MM, or MM:SS.S, whatever is appropriate.

Example: Date records might be consistently formatted with slashes, as MM/DD/YY.

Comment: The convention chosen should be that familiar to the prospective users. For European users, the formatting of telephone numbers and of dates is customarily different than suggested in the examples above. For military users, date/time data are frequently combined in a familiar special format. For many user groups, time records are kept on a 24-hour clock, which should be acknowledged in display formatting.

Reference: EG 2.2.17; MS 5.15.1.3.4.

-12 Long strings of arbitrary alphanumeric characters should be displayed in groups of three or four separated by a blank.

Exception: Words should, of course, be displayed intact, whatever their length.

Comment: Hyphens may be used instead of blanks where that is customary. Slashes are less preferred for separating groups, since they are more easily confused with alphanumerics.

Comment: Grouping should follow convention where a common usage has been established, as in the NNN-NN-NNNN of social security numbers.

Reference: BB 2.4.1; EG 2.2.2; MS 5.15.4.9.a.

2.1.3 Tables

-1 In tabular displays, columns and rows should be labeled following the same guidelines proposed for labeling the fields of data forms.

Reference: BB 2.8.7.

-2 In tabular displays, the units of displayed data should be included in the column labels.

Reference: BB 2.8.8.

-3 Columns of numeric data should be displayed right-justified, or justified with respect to a fixed decimal point.

Reference: BB 2.4.2, 2.4.3; EG 2.3.9; MS 5.15.4.9.d; PR 4.8.10, 4.10.6.

See also: 1.5-6.

-4 Lists of data should be vertically aligned with left justification to permit rapid scanning; indentation can be used to indicate subordinate elements in hierarchic lists.

Exception: Numbers should be right-justified.

Exception: A short list, of just 4-5 items may be displayed horizontally on a single line, in the interests of compact display format, if that is done consistently.

Reference: BB 2.3.1; EG 2.2.8, 2.2.11; MS 5.15.4.9.d, 5.15.4.9.e.

-5 Data lists should be organized in some recognizable order, whenever feasible, to facilitate scanning and assimilation.

Example: Dates may be ordered chronologically, names alphabetically.

Reference: EG 2.2.3, 2.3.1; MS 5.15.4.9.b.

-6 When listed data are labeled by number, letter, etc., the format chosen should be different than that used for selectable control options.

Reference: EG 2.2.7.

See also: 3.1.3-12.

2.1.3 Tables (cont.)

-7 When listed items are labeled by number, the numbering should start with "1", and not "0".

Comment: In counting, people start with "one"; in measuring, start with "zero".

Reference: EG 2.2.6.

-8 When data are displayed in more than one column, the columns should be separated by at least 3-4 spaces if right-justified, and by at least 5 spaces otherwise.

Reference: EG 2.3.6.

-9 Column spacing should be consistent from one display to another.

Reference: BB 2.8.3.

-10 When tables are used for referencing purposes, such as an index, the indexed material should be displayed in the left column, the material most relevant for user response in the next adjacent column, and associated but less significant material in columns further to the right.

Reference: Hamill, 1980.

-11 Items that must be compared on a character-by-character basis should be displayed with one directly above the other.

Reference: MS 5.15.4.6.2.d.

2.1.4 Graphics

-1 When users must scan and compare sets of data quickly, items should be displayed in an ordered graphic format, with backup display of raw data available as a user-selected option.

Comment: People cannot readily assimilate and compare detailed sets of raw data.

Reference: EG 2.2.9; MS 5.15.4.9.f.

2.1.4 Graphics (cont.)

-2 Graphic displays should be used, rather than alphanumeric, when a user must monitor changing data in any critical task involving qualitative distinction between normal and abnormal conditions.

Comment: It is preferable, of course, to program the computer to handle data monitoring, where that is feasible, and signal detected abnormalities to the user's attention.

Reference: Hanson, Payne, Shiveley and Kantowitz, 1981; Tullis, 1981.

-3 Graphic symbols should be standardized in meaning within a system, and among systems having similar operational requirements.

Reference: MS 5.15.2.6.

2.1.5 Combination

-1 When tables and/or graphics are combined with text, each figure should be placed immediately following its first reference in the text.

Comment: People may not look at a figure if it is displayed in a location separated from its reference.

Reference: Whalley and Fleming, 1975.

2.2 Data Selection

-1 When the user participates in selection of data for display, each display should have a unique identifying label, an alphanumeric code or abbreviation that can facilitate display requests by the user.

Comment: The display identification label should be short enough (3-7 characters) or meaningful enough to be remembered easily. Where flexibility is desired, it may be good practice to let each user assign names to the particular sets of data that constitute commonly used displays.

Reference: BB 2.1.1, 2.2.3.

2.2 Data Selection (cont.)

-2 The identifying label used for display selection should be displayed prominently in a consistent location.

Comment: The top left corner of the display is recommended for this purpose.

Reference: BB 2.2.3.

2.3 Data Aggregation

-1 Grouped data should be arranged in the display with consistent placement of items, so that user detection of similarities, differences, trends and relationships is facilitated.

Example:

Cost			Output		
Actual	Predicted	Difference	Actual	Predicted	Difference
947	901	+46	83	82	+ 1
721	777	-56	57	54	+ 3
475	471	+ 4	91	95	- 4

Reference: BB 2.8.6; Tullis, 1981.

-2 Displayed data should be grouped by some logical principle, such as sequence of use, where the spatial (or temporal) order is that in which data items are usually encountered.

Example: Data in an electronic display should match the order of items in an associated paper data form.

Reference: BB 2.8.1.

See also: 1.4-23, 1.4-24, 1.4-25.

-3 Displayed data should be grouped by some logical principle, such as by function, where data items associated with a particular question or purpose located adjacent to one another in the display.

Reference: BB 2.8.1; Tullis, 1981.

2.3 Data Aggregation (cont.)

-4 Displayed data should be grouped by some logical principle, such as importance, where data items providing the most significant information, and/or requiring immediate response, are grouped at the top of the display.

Reference: BB 2.8.1; Tullis, 1981.

-5 Displayed data should be grouped by some logical principle, such as frequency, where the most frequently used data items are presented at the top of the display.

Comment: Principles of data grouping also apply to the display/listing of control options.

Comment: These principles for data grouping in display formatting are essentially the same as those recommended for display/control layout in equipment design.

Reference: BB 2.8.1.

See also: 3.1.3-13.

-6 When there is no appropriate logic for grouping data by sequence, function, frequency or importance, some other principle should be adopted, such as alphabetical or chronological grouping.

Reference: BB 2.8.2.

See also: 2.1.1-14.

2.4 Display Generation

-1 System response to simple requests for data display should take no more than 0.5-1.0 second.

Example: This response time should apply when the user requests the next page of a multi-page display, or when a display begins to move in response to a scrolling request.

Comment: Responses to requests for new displays may take somewhat longer, perhaps 2-10 seconds, particularly if the user perceives such a request to involve more complicated operations, such as accessing different files, transforming data, etc.

Reference: EG Table 2.

-2 When displayed data are of potential long-term interest, there should be an easy means for the user to request local printing of a hard copy, within security restraints.

Comment: USI design should not require the user to rely on memory. Optional printout permits the user to record data from one display to compare with another, and so deal with situations where the system designer has not anticipated the need for such comparison.

Comment: The user should not have to take notes or transcribe displayed data manually. That practice under-utilizes the data handling potential of the computer, and risks transcription errors by the user.

Reference: BB 1.7; EG 4.2.14; MS 5.15.4.8; PR 4.10.1.

-3 The contents of a display should not change as a result of a user request for printout.

Reference: EG 4.2.14; MS 5.15.4.8.

2.5 Display Partitioning

-1 A consistent organization for the location of various display features should be adopted as a standard format to be used, insofar as possible, for all displays.

Example: One location might be used consistently for a display title, another area might be reserved for data output by the computer, and other areas dedicated to display of control options, instructions, error messages, and user command entry.

Comment: Consistent display formats are needed to establish and preserve user orientation. There is no fixed display format that is optimum for all data handling applications, which will vary in their requirements. However, once a suitable format has been devised, it should be maintained as a pattern to ensure consistent design of other displays.

Reference: BB 2.1, 2.8.4; EG 2.3, 2.3.3; MS 5.15.4.6.1.d.

-2 Means should be devised to make established display formats clearly perceptible to the user.

Example: Different display areas, or "windows", can be separated by spacing (where space permits); outlining can also be used to separate different areas, so that displayed data are distinct from control options, instructions, etc.

Reference: BB 2.8.5; EG 2.3; MS 5.15.4.6.2.

-3 Established display formats should be changed only as necessary to distinguish one task or activity from another.

Comment: The objective is to develop display formats that are consistent with accepted usage and existing user habits.

Reference: EG 2.2.5.

-4 The body of the display, when used for data output, should be formatted to present data coherently, and usually should <u>not</u> be partitioned into many small windows.

Reference: EG 2.3.2.

2.5 Display Partitioning (cont.)

-5 Every display should begin with a title or header, describing briefly the contents or purpose of the display; the title should be separated by one blank line from the body of the display.

Reference: BB 2.1.1, Table 1; PR 4.5.2.

-6 The last several lines at the bottom of every display should be reserved for status and error messages, prompts and command entry.

Comment: Assuming that the display is mounted physically above the keyboard, which is the customary placement, the user can look back and forth from keyboard to display more easily when prompts and entry area are at the bottom of the display.

Reference: BB 6.1.2; PR 4.5.3.

2.6 Display Density

-1 Ideally, each display should provide the user all of the information needed at that point in the transaction sequence.

Example: Header information should be retained, or re-generated, when paging /scrolling data tables.

Comment: The user should not have to remember information from one display to the next.

Reference: BB 4.3.4; EG 2.3.14, 2.3.15.

See also: 2.0-1, 2.8-1.

-2 Ideally, each display should provide the user only the information essential at that point in the transaction sequence, and not be overloaded with extraneous data.

Comment: Extraneous data will prevent or slow user assimilation of needed information. Where user information requirements cannot be accurately determined in advance of interface design, and/or are expected to be variable, on-line user options should be provided for data selection, display coverage and suppression.

Reference: BB 2.7, 2.8.10; EG 3.1.4; MS 5.15.2.3; Tullis, 1981.

See also: 2.0-1.

-3 For simple user-system dialogues, each line of a display should provide a single item of information.

Reference: PR 4.10.5.

2.7 Display Coding

-1 Important items requiring user attention should be highlighted on the display with some form of auxiliary coding, particularly when they appear infrequently.

Example: Such items might include recently changed data, or data discrepant exceeding acceptable limits or discrepant with some other defined criteria.

Comment: Position coding may suffice, i.e., always displaying important items in a particular location, as when an error message appears in a space otherwise left blank, but auxiliary codes may be needed as well.

Reference: EG 2.1.3, 2.3.12; MS 5.15.4.6.1.

-2 Display coding should also be considered for applications where the user must distinguish rapidly among different categories of displayed items, particularly when those items are distributed in an irregular way on the display.

-3 Alphanumeric characters can be used in effectively unlimited combinations, and should be considered for auxiliary coding in display applications where basic data presentation is not already alphanumeric (e.g., graphics).

Reference: EG Table 1.

-4 When using alphanumeric codes, a consistent convention should be adopted that all letters shall either be upper case or else lower case.

Comment: Computer logic should not distinguish between upper and lower case in user entry of codes.

Reference: BB 2.3.3.

-5 When codes combine letters and numbers, characters of each type should be grouped together rather than interspersed.

Example: Letter-letter-number ("HW5") will be read and remembered more accurately than letter-number-letter ("H5W").

Comment: Unfortunately, there are common instances in which this rule has been overlooked, such as the coding of English and Canadian postal zones.

Reference: BB 2.5.1.

2.7 Display Coding (cont.)

-6 Meaningful codes should be adopted in preference to arbitrary codes.

Example: A three-letter mnemonic code (DIR = directory) is easier to remember than a three-digit numeric code.

Reference: BB 3.6.2.

-7 Display (and entry) codes should be assigned so as to conform with population stereotypes, accepted abbreviations, and user expectations.

Example: Use M for "male", F for "female", rather than arbitrary digits 1 and 2. In color coding, use red for danger.

Reference: BB 2.3.5.

-8 When codes are assigned special meaning in a display, a definition should be provided at the bottom of the display.

Example: The legend on a map is a common example.

Comment: The definition should replicate the code, i.e., display the symbol, line width, etc., being defined. For a color code, each definition should be displayed in the appropriate color, e.g., "RED = hostile" in red.

Reference: BB 7.6.1.

-9 When arbitrary codes must be remembered by the user, they should be no longer than 4-5 characters.

Comment: When a code is meaningful, such as a mnemonic abbreviation or a word, it can be longer.

Reference: BB 2.5.2.

2.7 Display Coding (cont.)

-10 Symbols, and other codes as well, should be assigned to have consistent meanings from one display to another.

Comment: When coding is not consistent, the user's task of display interpretation may be made more difficult than if no auxiliary coding were used at all.

Reference: BB 3.6.1, 7.6.2.

See also: 2.1.1-24.

-11 Special symbols, such as asterisks, arrows, etc., should be considered for drawing attention to selected items in alphanumeric displays.

Comment: Symbols chosen for such an "alerting" purpose should not be used for other purposes in the display.

-12 When a special symbol is used to mark a word, it should be separated from the beginning of the word by a space.

Comment: A symbol immediately adjacent to the beginning of a word will impair legibility.

Reference: Noyes, 1980.

-13 Geometric shapes should be considered for discriminating different categories of data on graphic displays.

Comment: Approximately 15 different shapes can be distinguished readily. If that "alphabet" is too small, it may be possible to use component shapes in combination, as in some military symbol codes.

Reference: EG Table 1.

-14 When shape coding is used, the assignment of codes should be consistent for all displays, and based upon an established standard.

Comment: Although shape codes can often be mnemonic in form, their interpretation will generally rely on learned association as well as immediate perception. Existing user standards must be taken into account by the display designer.

Reference: MS 5.15.4.6.1.e.

2.7 *Display Coding* (cont.)

-15 A special form of shape coding, using lines of varying length, should be considered for applications involving spatial categorization in a single dimension.

Example: The length of a displayed vector might be used to indicate distance or speed.

Comment: Perhaps four lengths can be reliably distinguished in practical use. Long lines will add clutter to a display, but may be useful in special applications.

Reference: EG Table 1.

-16 A special form of shape coding, using lines of varying direction, should be considered for applications involving spatial categorization in two dimensions.

Example: The angle of a displayed vector might be used to indicate direction, i.e., heading or bearing.

Comment: Users can make fairly accurate estimates of angles for lines displayed at ten-degree intervals.

Reference: Smith, 1962a.

-17 When a line is added simply to mark or emphasize a displayed item, it should be placed under the designated item.

Comment: A consistent convention is needed to prevent ambiguity in the coding of vertically arrayed items; underlining is customary, and does not detract from word legibility.

Comment: For words from the Roman alphabet, underlining probably detracts from legibility less than overlining.

-18 Size coding, i.e., varying the size of displayed alphanumerics and other symbols, should be considered only for applications where displays are not crowded.

Comment: Perhaps as many as five sizes might be used for data categorization, but two or three will probably prove the practical limit except for printed displays.

Reference: EG Table 1; MS 5.15.4.6.1.e.

2.7 Display Coding (cont.)

-19 When size coding is used, a larger symbol should be at least 1.5 times the height of the next smaller symbol.

Reference: MS 5.15.4.6.1.e.

-20 Differences in brightness of displayed symbols should be considered for many display applications as a two-valued code.

Example: A data form might combine bright data items with dim labels to facilitate display scanning.

Comment: Perhaps as many as four brightness levels might be used, but at some risk of reduced legibility for the dimmer items.

Reference: EG Table 1; MS 5.15.4.6.1.b.

-21 When a capability for brightness inversion is available, i.e., where bright characters on a dark background can be changed under computer program control to dark on light (or vice versa), this should be considered for use in highlighting displayed items that require user attention.

Reference: PR 3.3.4.

2.7 Display Coding (cont.)

-22 Color coding should be considered for applications where the user must distinguish rapidly among several categories of data, particularly when data items are dispersed on the display.

Example: Different colors might be used effectively in a position display to distinguish friendly, unknown and hostile aircraft tracks, or to distinguish among aircraft in different altitude zones.

Comment: Color is a good auxiliary code, where a multi-color display capability is available. A color code can be overlaid directly on alphanumerics and other symbols without significantly obscuring them. Color coding permits rapid scanning and perception of patterns and relationships among dispersed data items.

Comment: Perhaps as many as 11 different colors might be reliably distinguished, or even more for trained observers; but as a practical matter it will prove safer to use no more than five or six.

Reference: BB 7.2; EG Table 1; MS 5.15.4.6.1.f; Smith, 1963a; Smith and Thomas, 1964; Smith, Farquhar and Thomas, 1965.

-23 Color coding should be used conservatively, with relatively few colors to designate critical categories of displayed data.

Comment: Arbitrary use of many colors may cause displays to appear "busy" or cluttered, and may reduce the likelihood that relevant color coding on other displays will be interpreted appropriately and quickly by the user.

Reference: BB 7.1.

-24 Color coding should be applied as an additional aid to the user on displays that have already been formatted as effectively as possible in a single color.

Comment: Do not use color coding in an attempt to compensate for poor display format; redesign the display instead.

Reference: BB 7.3.

2.7 Display Coding (cont.)

-25 When color coding is used, it should be redundant with some other feature in data display, such as symbology.

Comment: Displayed data should provide necessary information even when viewed at a monochromatic display terminal, or hard-copy printout, or when viewed by a user with defective color vision.

Reference: BB 7.4.

-26 When color coding is used, each color should represent only one category of displayed data.

Comment: Color will prove the dominant coding dimension on a display. If several different types of data are displayed in red, say, they will have an unwanted visual coherence that may hinder proper assimilation of information by the user.

Reference: BB 7.6.1; Smith and Thomas, 1964.

-27 Color coding should be consistent with conventional associations with particular colors.

Example: In a display of accounting data, negative numbers might be shown as red, corresponding to the customary use of red ink for that purpose.

Example: Red is associated with danger (in our society), and is an appropriate color for alarm conditions. Yellow is associated with caution, and might be used for alerting messages or to denote changed data. Green is associated with normal "go ahead" conditions, and might be used for routine data display. White is a color with neutral association, which might be used for general highlighting.

Comment: Other associations can be learned by the user if color coding is applied consistently.

Reference: BB 7.7.1, 7.7.2, 7.7.3; MS 5.15.4.6.1.f.

2.7 Display Coding (cont.)

-28 The color blue should be used only for background features in a display, and not for critical data.

Example: Blue might be used in shading background areas in graphic displays, where its lower apparent brightness could possibly be of benefit.

Comment: The human eye is not equally sensitive to all colors, nor are its optics color-corrected. Blue symbols appear dimmer than others, and are more difficult to focus.

Reference: BB 7.6, 7.7.5.

-29 Blink coding should be considered for applications where a displayed item implies an urgent need for user attention.

Comment: Blinking symbols are effective, if used sparingly, in calling the user's attention to displayed items of unusual significance. Blinking characters may have somewhat reduced legibility, and may cause visual fatigue with over-use.

Comment: Perhaps as many as four levels might be distinguished, but it will probably prove safer to use blinking as a two-level code, i.e, blinking versus non-blinking.

Reference: EG Table 1; MS 5.15.4.6.1.a; Smith and Goodwin, 1971b; Smith and Goodwin, 1972.

See also: 2.7-30.

-30 When blink coding is used to mark a data item that must be read, an extra symbol such as an asterisk should be added as a blinking marker, rather than blinking the item itself.

Comment: This practice will draw attention to an item without detracting from its legibility.

Reference: Smith and Goodwin, 1971b.

See also: 2.7-29.

2.7 Display Coding (cont.)

-31 When blink coding is used, the blink rate should be in the range of 2-3 Hz, with a minimum duty cycle (ON interval) of 50 percent.

Comment: Although equal ON and OFF intervals are often specified, an effective code can probably be provided even when the OFF interval is considerably shorter than the ON (perhaps a wink, rather than a blink), as in occulting lights used for Navy signaling.

-32 Other perceptual dimensions that should be considered for coding visual displays include line type (e.g., dashed versus solid), line width ("baldness"), focus, and motion.

Comment: Perhaps 3-4 line types might be readily distinguished, and 2-3 line widths. Only two levels of focus are feasible, clear and blurred, with the risk that blurred items will be illegible. Perhaps 2-10 degrees of motion might be distinguished, in display applications where motion is an appropriate and feasible means of coding.

Reference: EG 2.3.

-33 For auditory displays, distinctive sounds should be used to code items requiring special user attention.

Example: A variety of signals might be available, including sirens, bells, chimes, buzzers, and tones of different frequency.

Comment: Tones may be presented in sequence to enlarge the signal repertoire.

Reference: Smith and Goodwin, 1970.

-34 For auditory displays with voice output, different voices should be considered for use in distinguishing different categories of data.

Comment: At least two voices, male and female, could be readily distinguished, and perhaps more depending upon fidelity of auditory output, and listening conditions.

Reference: Smith and Goodwin, 1970.

2.8 Display Coverage

-1 Whenever possible, all data relevant to the user's current transaction should be included in one display frame.

Comment: Do not rely on the user to remember data accurately from one display to the next.

Reference: EG 3.4.4.

See also: 2.0-1, 2.6-1.

-2 When requested data exceeds the capacity of a single display frame, the user should be provided easy means to move back and forth among relevant displays, by paging or scrolling.

Example: Dedicated function keys might be provided for paging/scrolling forward and back.

Comment: Paging/scrolling is acceptable when the user is looking for a specific data item, but not when the user must discern some relationship among separately displayed sets of data.

Reference: BB 4.4.1, 4.4.2; EG 6.3.8; MS 5.15.4.9.j.

-3 When a list of numbered items exceeds one display frame, and must be paged/scrolled for its continuation, items should be numbered continuously in relation to the first item in the first display.

Reference: EG 2.3.10.

-4 When a tabular display must be paged/scrolled for its continuation, column headings and row labels should be preserved in each viewed portion of the display.

-5 When lists or tables are of variable length, and may extend beyond the limits of a single display frame, their continuation and ending should be explicitly noted on the display.

Example: Incomplete lists might be marked "continued on next page", or simply "continued". Concluding lists might add a note "end of list".

Exception: Short lists whose conclusion is evident from the display format need not be annotated in this way.

Reference: BB 2.9.6.

2.8 Display Coverage (cont.)

-6 When display output contains more than one page, the notation "page x of y" should appear on each display.

Comment: A recommended format is to put this note immediately to the right of the display title. With such a consistent location, the page note might be displayed in dimmer characters. Leading zeros should not be used in the display of page numbers.

Reference: PR 4.5.10, 4.10.4.

-7 When scrolling is used, a consistent orientation should be adopted in USI design as to whether 1) data are conceived to move behind a fixed display window, commonly called "scrolling"; or 2) the display window is conceived to move over a fixed array of data, here called "windowing".

Comment: A user can adapt to either concept, if it is maintained consistently. "Windowing" is the more natural concept for inexperienced users, causing fewer errors, and hence is the preferred option when other considerations are equal.

Reference: BB 4.4.8; Bury, Boyle, Evey and Neal (in press).

-8 In applications where a cursor is moved freely within a frame of displayed data, "windowing" should be selected rather than "scrolling" as the conceptual basis of display movement.

Example: full-screen editing.

Comment: Since displayed data will be perceived as fixed during cursor movement, considerations of joint compatibility suggest that displayed data remain conceptually fixed during window "movement". Indeed, it may be possible to use the same arrow-labeled keys to control both cursor movement and "windowing".

Reference: Morrill and Davies, 1961.

2.8 Display Coverage (cont.)

-9 When a "windowing" orientation is maintained consistently, the wording of scroll functions should refer to the display frame (or window) and not to the displayed data.

Example: The command "Up 10" should mean that ten lines of data will disappear from the bottom of the display, and ten earlier lines will appear at the top.

-10 When a "scrolling" orientation is maintained consistently, the wording of scroll functions should refer to the data being displayed, and not to the display frame or window.

Example: "Roll up 5 lines" should mean that the top five lines of data will disappear from the display, and five new lines will appear at the bottom.

Reference: EG 2.3.16.

-11 When the user may be exposed to different systems adopting different usage, any reference to scroll functions should avoid wording that implies spatial orientation, and instead consistently use functional terms such as "forward" and "back" (or "next" and "previous") to refer to movement within a displayed data set.

Comment: In that event, control of scroll functions should be implemented by keys marked with arrows, avoiding verbal labels altogether.

2.9 Display Update

-1 In accord with operational requirements, the user should be able to request automatic update (computer regeneration) of displayed data, and control the update rate.

-2 Changing data values that the user must read accurately should be updated only in a fixed position on the display, and no faster than one per second.

Reference: MS 5.15.4.5.1.

-3 Changing data values that the user need read only approximately to identify the general nature of data change should be updated no faster than five per second.

Reference: MS 5.15.4.5.2.

2.9 Display Update (cont.)

-4 Graphic displays in which a user must visually integrate changing patterns should be updated at a rate matching the user's data handling abilities.

Comment: Slowly developing patterns may be seen more easily with time compression, i.e., with rapid (and repetitive) display of sequentially stored data frames. Fast changing data may require time expansion, i.e., slowed motion, to detect patterns.

Comment: Similar considerations may apply to auditory displays, where speeding or slowing sound signals may aid pattern recognition.

Reference: MS 5.15.4.5.3.

-5 When a display is automatically updated, the user should be able to stop the process ("freeze", "stop action") at any point, to examine changed data more deliberately.

Comment: For some applications, it may also prove helpful if the user can step incrementally forward or back in the time sequence, frame by frame.

Reference: MS 5.15.4.5.4.

-6 When an updated display is frozen, some appropriate label should be added to remind the user of that status.

Reference: MS 5.15.4.5.5.

-7 When a display being updated in real time is frozen, the user should be warned if some significant (but not displayed) change is detected in the computer processing of new data.

Reference: MS 5.15.4.5.4.

-8 When a display being updated in real time is frozen, resumption of display update should be at the current real-time point unless otherwise specified by the user.

Comment: In some applications, a user might wish to resume display update at the point of stoppage, and so display change would thenceforth lag real-time data change. As a general practice, however, such an option risks confusion.

Reference: MS 5.15.4.5.4.

2.10 Display Suppression

(no guidelines presently available)

2.11 Design Change

(no guidelines presently available)

APPENDIX D

DESIGN GUIDELINES FOR SEQUENCE CONTROL FUNCTIONS

Sequence control refers to the logic and means by which inputs and outputs are linked to become coherent transactions, and which govern the transitions from one transaction to the next. Techniques of sequence control require explicit attention in USI design, and a number of published guidelines bear on this topic.

A fundamental decision in USI design is selection of the dialogue type(s) that will be used to implement sequence control. Here dialogue refers to the sequence of transactions which mediate user-system interaction. Ramsey and Atwood (1979, pp. 76-95) identify eight general dialogue types:

question and answer
form filling
menu selection
function keys
command language
query language
natural language
interactive graphics

Various sub-categories can be identified, of course, as illustrated by the many examples in Martin's book on the subject (1973).

USI design will often involve a mixture of two or more dialogue types, since different dialogues are appropriate to different jobs and different kinds of users. Recognition of appropriate dialogue types at the outset of system development will facilitate USI design and help ensure the effectiveness of future system operation.

The selection of dialogue types based on anticipated task requirements and user skills seems straightforward, at least for simple cases. Computer-initiated question-and-answer dialogues are suited to routine data entry tasks, where data items are known and their ordering can be constrained, and provides explicit prompting for unskilled, occasional users. Form-filling dialogues permit somewhat greater flexibility in data entry, but may require user training. When data entries must be made in arbitrary order, perhaps mixed with queries as in making airline reservations, for example, then some mixture of function keys and coded command language will be required for effective operation, implying a moderate to high level of user training.

From these examples, it would seem possible to judge for any information handling task the types of dialogue that should prove most suitable, and to indicate this judgment in the USI functional capabilities checklist in specification of USI requirements.

One important aspect of dialogue choice is that different types of dialogue imply differences in system response time for effective operation. In a repetitive form-filling dialogue, for example, the user may accept relatively slow computer processing of a completed form. If the computer should take several seconds to respond, the user probably can take that time to set one data sheet aside and ready another. But several seconds delay in a menu selection dialogue may prove intolerable, especially where the user must make an extended sequence of selections in order to accomplish a desired end.

To categorize these differences, Table D-1 presents for each of the eight general dialogue types identified above an estimate of the implied requirement for user training and for system response time. Cumulative experience and specific requirements of a particular task may modify such estimates. But the general principle illustrated here, that one design choice implies others, must be taken into account in USI specification.

Table D-1

Estimated Requirements of Different Dialogue Types

Dialogue Type	Required User Training	Required System Response Time
Question and Answer	Little/None	Moderate
Form Filling	Moderate/Little	Slow
Menu Selection	Little/None	Very Fast
Function Keys	High/Moderate	Fast
Command Language	High	Fast
Query Language	High/Moderate	Moderate
Natural Language	Moderate (potentially Little)	Fast
Interactive Graphics	High	Very Fast

Flexibility and context are important in sequence control as in other aspects of USI design. Ideal flexibility would permit the user to undertake whatever task or transaction he wishes, at any time. Although this may not always prove feasible, the USI designer should try to provide the maximum possible user control of the on-line transaction sequence. As a simple example, suppose the user is scanning a multi-page data display. He should be able to go either forward or back at will. If the USI design only permits him to step forward, so that he must cycle through the entire display set to reach a previous page, that design is inefficient. The user should also be able to interrupt display scanning at any point to initiate some other transaction. Such simple flexibility is relatively easy for the designer to achieve, and indeed is commonly provided.

More difficult are transactions that involve potential change to stored data. Here again the user will need flexibility in sequence control, perhaps wishing to back up in a data entry sequence to change previous items, or to cancel and restart the sequence, or to abort the sequence altogether and escape to some other task. The USI designer can provide such flexibility through use of suspense files and other special programmed features. This flexibility requires extra effort from the designer and programmer. But that extra effort is made only once, and is a worthwhile investment on behalf of future users who may interact with their computer system for months or even years.

In one respect, flexibility of sequence control has pitfalls. Just as a user can make a mistake in data entry, so also can he make a mistake in sequence control. The USI designer must try to anticipate user errors and ensure that potentially irreversible actions are difficult to take. In data entry tasks, for example, when a user is satisfied with a set of data he should be obliged to take some explicit action to ENTER it for computer processing. The USI should be designed to protect the user from the consequences of inadvertently destructive actions. Any large-scale erasure or deletion of data, for example, should require some sort of explicit user confirmation, being accomplished as a two-step process rather than a single transaction. (This provides a software analogy to the physical barriers sometimes used to protect critical hardware controls from accidental activation.) Some well-designed systems go a step further, and permit the user to reverse a mistaken action already taken.

One form of flexibility frequently recommended is the provision of alternate modes of sequence control for experienced and inexperienced users. In a command-language dialogue, optional guidance might be provided to prompt a beginner step by step in the

composition of commands, whereas an experienced user might enter a complete command as a single complex input. Some such flexibility in USI design is surely desirable -- to interpret halting, stepwise control inputs, as well as fluent, coherent commands.

More generally, however, it may be desirable to include redundant modes of sequence control in USI design, perhaps involving combinations of different dialogue types. As an example, menu selection might be incorporated to provide easy sequence control for beginners, but every display frame might also be formatted to include a standard field where an experienced user could enter complete commands more efficiently. Examples of this approach have been provided by Palme (1979).

Another way to provide flexibility in sequence control is through specific tailoring of display formats. Consider, for example, a dialogue type in which sequence control is exercised through lightpen selection among displayed command options. For any particular display frame it might be possible to display just three or four options most likely to be selected by a user at that point in the task sequence, plus a general purpose OPTIONS selection that could be used to call out a display of other (less likely) commands. Thus, on the first page of a two-page display set, one of the likely commands would be NEXT PAGE; but on the second page that command would be replaced by its more likely complement, PREV PAGE.

This approach illustrates two design ideas. The first comes close to being a general rule for sequence control: make the user's most frequent transactions the easiest to accomplish. The second idea is the reliance on context to improve flexibility.

The importance of context in sequence control extends beyond the example above, where likely control options are given preferential display. Context can be used to shorten command inputs during sequence control and still permit unambiguous interpretation. In general, it will prove desirable to have the user rather than the computer define the task context in which control inputs are to be interpreted.

As an example, suppose that a user wishes to assign several sorties of aircraft from a particular squadron to preplanned missions. He should be able to specify the squadron and then expect all subsequent commands to be applied to that squadron. Further, he should be able to specify that he is making assignments and expect that subsequent selections of aircraft/crews and missions will be interpreted accordingly. If context can be used in this way, the user can avoid repetitious control actions, which is just as desirable as avoiding repetitious data entries.

A potential pitfall here is that the results of a particular control action, if contingent on context, may vary from one time to another. Thus aircraft selection in the context described above would result in mission assignment, whereas aircraft selection in a different context might result in unassignment, or allocation to ready reserve, or commitment to maintenance status. Such variability may prove confusing to the user. When the consequences of control actions are contingent on context, then the USI designer should ensure that the current context and its implied contingencies are always displayed to the user. That will help provide needed user guidance.

These general ideas concerning sequence control are reflected in the specific design guidelines proposed in the following pages.

SEQUENCE CONTROL Objectives:

Minimized control actions by user
Low memory load on user
Consistency of control actions
Compatibility with user needs
Flexibility of sequence control

3.0 General

-1 Flexible means of sequence control should be provided so that the user can accomplish necessary transactions involving data entry, processing, retrieval and transmission, or can obtain guidance as needed in connection with any transaction.

Example: In scanning a multi-page display the user should be able to go forward or back at will; if USI design permits only forward steps, so that the user must cycle through the entire display series to reach a previous page, that design is deficient.

Comment: Necessary transactions should be defined in task analysis prior to software design.

Reference: PR 4.0.

-2 Control inputs should be simplified to the maximum extent possible, particularly for real-time tasks requiring fast user action, and should permit completion of a transaction sequence with the minimum number of control inputs consistent with user abilities.

Example: The user should be able to print a display directly without having to take a series of other actions first, such as calling for the display to be filed, specifying a file name, then calling for a print of that named file.

Example: For long, multi-page displays, it should be possible to request a particular page directly, without having to take repetitive PAGE FORWARD actions.

Comment: Shortcuts via direct commands should be provided for experienced users, to by-pass intervening steps that might provide a more easily learned sequence for beginners. The computer should be programmed to handle intervening steps automatically, informing the user what has been done if that seems necessary.

Reference: BB 4.4.1, 4.5; MS 5.15.2.7.

3.0 General (cont.)

-3 The means of sequence control should be compatible with desired ends; frequent or urgent control actions should be easy to take, whereas potentially destructive control actions should be made sufficiently difficult to require explicit user attention.

Comment: Perhaps "difficult" is the wrong word here. If a destructive action is made different or distinctive in some way, so that it will not be taken by mistake, then perhaps it is not necessary that it be made difficult as well.

Reference: EG 4.0, 4.1.2.

-4 Sequence control should be compatible with user skills, permitting simple step-by-step control actions for beginners, and efficiently coded command entry by experienced users.

Comment: This will generally require a mix of dialogue types.

See also: Section 3.1.

-5 In most on-line information handling systems, sequence control should result from explicit user inputs rather than occur as an automatic consequence of computer processing.

Example: The computer should not interrupt user data entry to require immediate correction of any input error, but instead should wait until the user signals completion of the transaction.

Exception: Routine, repetitive transaction sequences in which successful completion of one may lead automatically to initiation of the next.

Exception: Automated process control applications where emergency conditions may take precedence over current user transactions.

Comment: In general, computer detection of problems with current user inputs can be negotiated at the conclusion of a transaction, before it is implemented. Computer detection of other problems can be signaled by alarms or advisory messages, so that the user can choose when to deal with them.

See also: 1.0-8, 1.1-5, 1.4-1.

3.0 General (cont.)

-6 Although the user-system dialogue is necessarily limited by the computer, software design should insofar as possible permit initiative and control by the user; the USI designer should anticipate all possible user actions and their consequences, and should provide appropriate options in every case.

Comment: In particular, a dialogue should never reach a dead end with no further action available to the user; if the user makes an input inappropriate (or unrecognizable) to current processing logic, the result should simply be an advisory message indicating the nature of the problem and the available options as to what can be done next.

Reference: PR 2.2.

-7 Whenever possible, control inputs should be self-paced, depending upon the user's needs, attention span and time available, rather than computer processing or external events.

Comment: When self-pacing does not seem feasible, the general approach to task allocation and USI design should be reconsidered.

See also: 1.0-6.

-8 The speed of computer response to user inputs should be appropriate to the transaction involved; in general, the response should be faster for those transactions perceived by the user to be simple.

Example: Computer response to a predictable command, such as NEXT PAGE, should be within 0.5-1.0 second; response to other simple commands should be within 2.0 second; error messages should be displayed within 2-4 second.

Reference: Miller, 1968.

-9 Control inputs by a user should not be delayed or paced by delays in computer response.

Comment: It is recommended that control delays or lockouts not exceed 20 milliseconds. In some applications, however, longer delay may be tolerable, particularly if that has the effect of reducing variability in computer response time.

Reference: MS 5.15.3.3.

See also: 1.0-7.

3.0 General (cont.)

-10 If further user inputs must be delayed pending completion of computer processing, the keyboard should be automatically locked until the user can begin a new transaction; keyboard lock should be accompanied by disappearance of the cursor from the display and (especially if infrequent) by some more specific indicator such as an auditory signal.

Comment: Absence of a cursor is not in itself a sufficient indicator of keyboard lockout. Auditory signals will be particularly helpful to skilled touch typists, who may not look at the display during a repetitive data transcription.

Comment: Following keyboard lockout, computer readiness to accept further inputs should be signaled to the user.

Comment: In some cases, it may be desirable to provide the user with an auxiliary means of control input, such as a special function key, to abort a transaction causing extended keyboard lockout.

-11 When execution of a control input is delayed, the computer should give the user some positive indication when processing is subsequently completed, the outcome, and the implied need for further user actions if any.

Reference: BB 4.3.1; MS 5.15.1.4.c.

-12 All control inputs made by a user should be acknowledged unambiguously by the computer, either by their immediate execution, or else by some immediate message indicating that execution is in progress or deferred or that the control input requires correction or confirmation.

Example: In particular, the absence of computer response is not an acceptable means of indicating that a command is being processed.

Comment: "Immediate" as used here is subject to interpretation in relation to the response time requirements of different dialogue types.

Reference: BB 4.3.2; EG 4.2.5; MS 5.15.3.2, 5.15.3.4.

See also: Section 3.1.

3.0 General (cont.)

-13 Computer responses to control inputs should generally consist of changes in state or value of displayed elements affected by the control action, in an expected or natural form.

Reference: MS 5.15.3.4.1.

-14 In data entry tasks where input is usually accomplished as a single, discrete transaction, successful entry should be signaled by a confirmation message without removing any visual display of the entered data.

Comment: This follows the general recommendation for sequence control that the user should leave one transaction and choose the next by explicit action.

Reference: MS 5.15.1.2.7.d.

-15 In data entry tasks where input is repetitive, in a continuing sequence of transactions, successful entry should be signaled by regeneration of the data entry display, automatically removing the just entered data in preparation for the next entry.

Comment: This represents an exception to the general principle of sequence control by explicit user choice, in the interest of efficiency.

Comment: An explicit message confirming successful data entry can be added in cases where that seems helpful.

Reference: EG 4.2.10.

-16 Sequence control actions should be consistent in form and consequences throughout USI design; similar means should be employed to accomplish similar ends, from one transaction to the next, and from one task to another.

Comment: In particular, there should be some standard, consistent routine for the user to initiate and complete task sequences.

3.0 General (cont.)

- If the consequences of a given control action will differ depending upon context established by a prior action, then some appropriate means of context definition should be displayed in advance to the user.

  Comment: Do not rely on the user always to remember prior actions, nor to understand their current implications.

-18 The design of linked transaction sequences should be based on task analysis, i.e., should represent a logical unit or subtask from the viewpoint of the user.

  Comment: A logical unit to the user is not necessarily the same as a logical unit of the computer software that mediates the transaction sequence.

  Reference: PR 5.1.

-19 Displays should be designed so that features relevant to sequence control are distinctive in position and/or format.

  Comment: Relevant features include displayed options, any command entry area used to indicate control actions, prompts, advisory messages, and other displayed items (titles, time signals, etc.) whose changes signal the results of control actions.

-20 When two or more users must interact with the system simultaneously, control inputs by one should not interfere with those of another.

  Comment: This requires careful USI design for applications where joint, coordinated actions must be made by a group of users.

  Reference: MS 5.15.2.5.

-21 In instructional material, and in on-line messages to the user, consistent terminology should be adopted to refer to control inputs.

  Example: Various words and phrases might be used, such as "control input", "command entry", "instruction", "request", "function call", etc. The practice adopted in these guidelines is to call general sequence control actions "control inputs" and to call control inputs keyed onto the display "command entries".

3.1 Dialogue Type

-1 Choice of dialogue type(s) and design of sequence control dialogue should take into account user characteristics and task requirements.

Example: When data entries must be made in arbitrary order, perhaps mixed with queries (as in making flight reservations), then some mixture of function keys and coded command entries will be required for effective operation, implying a moderately high level of user training.

Comment: The simple dictum is "Know the user". If user characteristics are variable, which is often the case, then a variety of dialogue types should be provided.

-2 The speed of computer response to user inputs should be appropriate to the type of dialogue; in general, the response to menu selections, function keys, and most inputs during graphic interaction should be immediate.

Comment: It is generally thought that maximum acceptable computer response for menu selection by lightpen is 1.0 second; for key activation is 0.1 second; for cursor positioning by lightpen (as in graphic line drawing) 0.1 second.

Comment: If computer response time will be slow, other dialogue types should be considered by the USI designer.

Reference: Miller, 1968.

3.1.1 Question and Answer

-1 Question-and-answer dialogue should be considered primarily for routine data entry tasks, where data items are known and their ordering can be constrained, where the user will have little or no training, and where computer response is expected to be moderately fast.

Comment: Brief question-and-answer sequences can be used to supplement other dialogue types for special purposes, such as for log-on routines, or for resolving ambiguous control inputs or data entries.

3.1.2 Form Filling

-1 Form-filling dialogue should be considered when some flexibility in data entry is needed, such as the inclusion of optional as well as required items, where users will have moderate training, and/or where computer response may be slow.

See also: Section 1.4.

3.1.3 Menu Selection

-1 Menu selection should be considered for tasks such as scheduling and monitoring that involve little entry of arbitrary data, where users may have relatively little training, and where computer response is expected to be fast.

Comment: Menu selection is, of course, a generally good means of mediating control inputs by untrained users, in conjunction with other dialogue types for other task requirements.

Comment: When display output is slow, as for a printing terminal, or for an electronic display constrained by a low-bandwidth channel, it may be tiresome for a user to wait for display of menu options.

-2 Each menu display should require just one selection by the user.

Comment: Beginning users will be confused by any more complicated "Chinese menu" requiring one choice from Column A, two from Column B, etc.

Reference: PR 4.6.5.

-3 When menu selection is the primary means of sequence control, and especially if extensive lists of control options must be displayed, then selection should be accomplished by direct pointing (e.g., by lightpen).

See also: 1.1-14.

3.1.3 Menu Selection (cont.)

-4 If menu selection is handled by pointing, dual activation should be provided, the first action to designate (position a cursor on) the selected option, followed by a separate action to make an explicit control input.

Comment: The two actions should be compatible in their design implementation. If the cursor is positioned by keying, then an ENTER key should be used to signal control input. If the cursor is positioned by lightpen, then a dual-action "trigger" on the lightpen should be provided for positioning and control inputs.

Comment: This recommendation assumes that accuracy in selection of control inputs is more important than speed. In some applications that may not be true. USI design will involve a trade-off considering the criticality of wrong inputs, ease of recovery from wrong inputs, and user convenience in making selections.

See also: 1.0-8, 3.0-5.

-5 When menu selection is a secondary (occasional) means of control input, and/or only short option lists are needed, then selection may be accomplished by keyed entry of corresponding codes, or by other means such as programmed multi-function keys labeled in the display margin.

-6 When menu selection is accomplished by code, that code should be keyed into a standard command entry area (window) in a fixed location on all displays.

Example: In a customary terminal configuration, with the display located above the keyboard, command entry should be in the bottom line of the display.

Comment: In effect, the command entry area should be positioned to minimize user head/eye movement between the display and the keyboard.

Comment: For experienced users, coded menu selections can be keyed in a standard area identified only by its consistent location and use; if the system is designed primarily for novice users, that entry area should be given an appropriate label such as ENTER CHOICE HERE: ___.

Reference: MS 5.15.4.6.1.d; PR 4.6.3.

3.1.3 Menu Selection (cont.)

-7 Menu options should be worded so as to permit direct selection of any option as an acceptable control input, either by pointing or by code entry; options should not be worded so as to imply a question requiring a YES/NO answer.

Example: +PRINT is acceptable; PRINT? is not.

Reference: PR 4.6.8.

-8 When control inputs will be selected from a discrete set of options, then those options should be displayed at the time of selection.

Reference: MS 5.15.3.5.

-9 If menu selection is used in conjunction with (as an alternative to) command language, then displayed control options should be worded in terms of recognized commands or command elements.

Comment: Where appropriate, sequences of menu selections should be displayed in an accumulator until the user signals entry of a completely composed command.

Comment: This practice will speed the transition for a novice user, relying initially on sequential menu selection, to become an experienced user composing coherent commands without such aid.

3.1.3 Menu Selection (cont.)

-10 If menu selections must be made by keyed codes, options should be coded by the initial letter (or letters) of their displayed labels, rather than by more arbitrary numeric codes.

Exception: Options might be numbered when a logical order or sequence is implied.

Exception: When option selection is from a long list, line number might be an acceptable alternative to letter codes.

Comment: Letters are easier than numbers for touch typists; options can be re-ordered on a menu without changing letter codes; it is easier to memorize meaningful names than numbers, and so letter codes can facilitate a potential transition from menu selection to command language when those two dialogue types are used together.

Comment: USI designers should not create unnatural option labels just to ensure that the initial letter of each will be different. There must be some natural difference among option names, and a two- or three-letter code can probably be devised to emphasize that difference.

Reference: Palme, 1979. (Note contrary views, favoring number codes, in BB 2.9.3; EG 2.2.7; PR 4.6.2.)

-11 If letter codes are used to make menu selections, then insofar as possible those codes should be used consistently in designating options at different steps in a transaction sequence.

Example: The same action should not be given different names and hence different codes (F = FORWARD and N = NEXT); the same code should not be given to different actions (Q = QUIT and Q = QUEUE).

-12 If menu options are included in a display intended also for data review and/or data entry, which is often a practical design approach, the labels for control input should be located consistently in the display and should incorporate some consistent distinguishing feature to indicate their special function.

Example: All control options might be displayed beginning with a special symbol, such as a plus sign (+FORWARD, +BACK, etc.)

See also: 2.1.3-6.

3.1.3 Menu Selection (cont.)

-13 Displayed menu options should be listed in a logical order; if no logical structure is apparent, then options should be displayed in order of their expected frequency of use, with the most frequent listed first.

Comment: If the first menu option is always the most likely choice, then for some applications it may be useful for efficiency of sequence control if a null input defaults to the first option. If that is done, it should be done consistently.

Reference: BB 2.9.4; PR 4.6.6; Palme, 1979.

See also: 2.3-5.

-14 Displayed menu lists should be formatted to indicate the hierarchic structure of logically related groups of options, rather than as an undifferentiated string of alternatives.

Comment: When logical grouping requires a trade-off against expected frequency of use, USI designers should resolve that trade-off consistently throughout the menu structure.

-15 If menu options are grouped in logical subunits, those groups should be displayed in order of their expected frequency of use.

Reference: PR 4.6.6.

-16 If menu options are grouped in logical subunits, each group should be given a descriptive label that is distinctive in format from the labels of the control options themselves.

Comment: Although this practice might sometimes seem to waste display space, it will help provide user guidance; moreover, careful selection of group labels may serve to reduce the number of words needed for individual option labels.

Reference: MS 5.15.2.10.

3.1.3 Menu Selection (cont.)

-17 A displayed menu should include only options appropriate at that particular step in a transaction sequence, and for the particular user.

Example: Displayed file directories should contain only those files actually available to the user.

Example: An UPDATE option should be offered only if the user has update rights for the particular data file being used.

Exception: Menu displays for a system still under development might indicate future options not yet implemented, but those options should be specially designated in some way.

Comment: A seeming exception might be a process control display in which current values of a number of variables must be monitored (i.e., must be displayed continuously), and where supplementary data (e.g., trend analysis) can be called out for some variables but not others. Here some means must be found to signal the user which variables can be selected and which not.

-18 Insofar as possible a displayed menu should include all options appropriate at that particular step in a transaction sequence.

Exception: A familiar set of general control options always available may be omitted from individual displays, and accessed as needed by a +OPTIONS input.

See also: Section 3.2.

3.1.3 Menu Selection (cont.)

-19 When option selections must be made from a long list, and not all options can be displayed at once, a hierarchic sequence of menu selections should be provided rather than one long multi-page menu.

Exception: Where a long list is already structured for other purposes, such as a list of customers, a parts inventory, a file directory, etc., it might be reasonable to require the user to scan multiple display pages to find a particular item. Even in such cases, however, an imposed structure for sequential access may prove more efficient.

Comment: Beginning users may prefer a menu permitting a single choice from all available options, when those can be displayed on one page. Experienced users, however, may perform faster with a sequence of choices from a hierarchy of separately displayed sub-menus.

Comment: A single menu that extends for more than one page will hinder learning and use. The USI designer can usually devise some means of logical segmentation to permit several sequential selections among few alternatives instead of a single difficult selection among many.

Reference: Dray, Ogden and Vestewig, 1981.

-20 When the user must step through a sequence of menus to make a selection, the hierarchic structure should be designed, insofar as possible within the constraints of display space, to minimize the number of steps required.

Comment: This represents a trade-off against the previous guideline. The number of hierarchic levels should be minimized, but not at the expense of display crowding.

Comment: When space permits, it may be desirable to display further (lower) choices in the hierarchic structure, to give the user a deeper view of the structure and permit direct selection of specific lower-level options.

Reference: MS 5.15.2.2.a; Miller, 1981.

3.1.3 Menu Selection (cont.)

-21 When hierarchic menus are used, they should be designed to permit the user immediate access to critical or frequently selected options.

Reference: MS 5.15.2.2.b.

-22 When hierarchic menus are used, the user should be given some displayed indication of current position in the menu structure.

Comment: One possible approach would be to recapitulate prior (higher) menu selections on the display. If routine display of path information seems to clutter menu formats, then such information might be provided only as an optional display at user request.

Reference: MS 5.15.2.2.c.

-23 When hierarchic menus are used, care should be taken to ensure consistent display formats at each level.

Reference: MS 5.15.2.2.d.

-24 When hierarchic menus are used, a single key action should permit the user to return to the next higher level.

Reference: BB 4.4.4.

-25 Menus provided in different displays should be designed so that option lists are consistent in terminology and ordering.

Example: If +PRINT is the last option in one menu, the same print option should not be worded +COPY at the beginning of another menu.

-26 When a control option has been selected and entered, if there is no immediately observable natural response some other form of acknowledgment should be displayed.

Comment: An explicit message might be provided. In some applications, however, it may suffice simply to highlight the selected option label (e.g., by brightening or inverse video) when that would provide an unambiguous computer acknowledgment.

Reference: MS 5.15.1.4.a.

See also: 1.1-6.

3.1.3 Menu Selection (cont.)

-27 Experienced users should be provided means to by-pass a series of menu selections and make an equivalent command entry directly.

Reference: BB 6.7.

See also: 3.0-2.

-28 When a user can anticipate menu selections before they are presented, means should be provided to enter several "stacked" selections at one time.

Comment: If necessary, stacked sequential entries might be separated by a special character, such as a slash, comma or semicolon. It would be preferable, however, if they could simply be strung together without special punctuation.

Reference: BB 6.8.

3.1.4 Function Keys

-1 Function keys should be considered for tasks requiring only a limited number of control inputs, or in conjunction with other dialogue types as a ready means of accomplishing critical inputs which must be made quickly without syntax error.

Reference: MS 5.15.3.7.

-2 Function keys should be considered as a means of accomplishing frequently required control inputs.

Example: ENTER, PRINT, PAGE FORWARD, PAGE BACK, OPTIONS, etc.

Comment: When generally used options are always implicitly available via function keys, they need not be included in displayed menus.

Reference: BB 4.4.

See also: 3.1.3-18, Section 3.2.

3.1.4 Function Keys (cont.)

-3 Function keys should be used as a means of permitting interim control inputs, i.e., for control actions taken before the completion of a transaction.

Example: TAB, DITTO, DEFAULT, HELP, etc.

-4 Function keys should be labeled informatively to designate the function they perform; labels should be sufficiently different from one another to prevent user confusion.

Example: Log-on should not be initiated by a key labeled PANIC. (This example may seem unlikely, but is cited from an actual USI design.)

Example: Two keys should not be labeled ON and DN.

Reference: BB 4.4.5; MS 5.15.2.10.

-5 If a function is continuously available, to serve a single function, just one label should be on the key.

-6 If a function key is used for more than one function, the user should always have some convenient means of knowing which function is currently available.

Comment: If a key is used for just two functions, depending upon defined operational mode, then alternate self-illuminated labels should be provided on the key to indicate which function is current. In these circumstances, it is preferable that only the currently available function is visible, so that the labels on a group of keys will show what can be done at any point rather than what has been done.

Comment: If the function of a key is specific to a particular step in the transaction sequence, then the current function should be indicated by an appropriate guidance message on the user's display.

Reference: MS 5.15.3.8.

-7 If a function is assigned to a particular key for one task/transaction, that function should be assigned consistently to the same key in other transactions.

Reference: BB 4.4.5.

3.1.4 Function Keys (cont.)

-8 When a function key performs different functions in different operational modes, those functions should be made as consistent as possible.

Example: A key labeled RESET should not be used to dump data in one mode, save data in another, and signal task completion in a third.

-9 When the set of functions assigned to keys changes as a result of user selection (so-called multifunction keys), an easy means should be provided for the user to return to the initial, base-level functions.

Comment: In effect, multifunction keys can provide hierarchic levels of options much like menu selection dialogues, with the same need for rapid return to the highest-level menu.

Comment: For some applications, it may be desirable to automate the return to base-level assignment of multi-function keys, to occur immediately on completion of a transaction and/or by time-out following a period of user inaction. The optimum period for any automatic time-out would have to be determined empirically for each application.

Reference: Aretz and Kopala, 1981.

-10 Function keys (and other devices) not needed for current inputs should be temporarily disabled under computer control at any step in a transaction sequence; mechanical overlays manipulated by the user should not be used for this purpose.

Comment: If the user selects a function key that is invalid at a particular step in a transaction sequence, no action should result except display of an advisory message indicating what functions are available at that point.

Reference: MS 5.15.3.9.4.3; PR 4.12.4.5.

3.1.4 Function Keys (cont.)

-11 When some function keys are active and some are not, the current subset of active keys should be indicated in some noticeable way to the user, perhaps by brighter illumination.

Comment: This practice will speed user selection of function keys.

Reference: Hollingsworth and Dray, 1981.

-12 Function keys should be grouped in distinctive locations on the keyboard to facilitate their learning and use; frequently used function keys should be placed in the most convenient locations.

Comment: It is preferable that frequently used keys not require double (control/shift) keying.

Reference: MS 5.15.4.6.1.d.

-13 The layout of function keys should be compatible with their importance; keys for emergency functions should have a prominent position and distinctive coding (e.g., size and/or color); keys with potentially disruptive consequences should be physically protected.

-14 Function keys should require only single activation to accomplish their function, and should not change function with repeated activation.

Example: Log-on should not be initiated by pressing a PANIC key twice.

-15 When function key activation does not result in any immediately observable natural response, the user should be given some other form of computer acknowledgment.

Comment: Temporary illumination of the function key would suffice, if key illumination is not used to signal available options. Otherwise a displayed advisory message should be used.

3.1.5 *Command Language*

-1 Command language dialogue should be considered for tasks involving a wide range of user inputs, where users may be highly trained in the interests of achieving efficient performance, and where computer response is expected to be relatively fast.

Comment: Command language should also be considered for data entry in arbitrary sequence.

-2 When command language is used for control input, an appropriate entry area should be provided in a consistent location on every display, preferably at the bottom.

Comment: Adjacent to the command entry area there should be a defined display window used for prompting control input, for recapitulation of command sequences (with scrolling to permit extended review), and to mediate question-and-answer dialogue sequences (i.e., prompts and responses to prompts).

Reference: MS 5.15.4.6.1.d.

-3 The words chosen for a command language should reflect the user's point of view and not the programmer's, corresponding consistently with the user's operational language.

Reference: EG 4.1.1, 4.2.12, 4.2.13; MS 5.15.1.4.5.

3.1.5 Command Language (cont.)

-4 Abbreviation of entered commands (i.e., entry of the first 1-3 letters) should be permitted to facilitate entry by experienced users.

Example: If a "P" uniquely identifies a print command (i.e., no other commands start with "P") then the user should be able to enter PRINT, or PR, or P, or any other truncation to initiate printing.

Comment: As a corollary, misspelling of command entries should also be tolerable, within the limits of computer recognition. The computer can interrogate the user as necessary to resolve ambiguous entries.

Comment: Variable abbreviation, i.e., keying only enough characters of a command to uniquely identify it, should probably not be used when the command set is changing. For the user, an abbreviation that works one day may not work the next. For the programmer, the addition of any new command may require software revision of recognition logic for other commands.

Reference: BB 6.4.3; Demers, 1981.

-5 All words in a command language, and their abbreviations, should be consistently used and standardized in meaning from one transaction to another, and from one task to another.

Example: Do not use EDIT in one place, MODIFY in another, UPDATE in a third, all referring to the same kind of action.

Reference: EG 4.2.9, EG 4.2.13; MS 5.15.2.6; Demers, 1981.

-6 Words in a command language should be chosen to be distinctive from one another, and to emphasize significant differences in function, in order to minimize user confusion.

Example: Do not label two commands DISPLAY and VIEW, when one permits editing displayed material and one does not.

Comment: In general, do not give different commands semantically similar names, such as SUM and COUNT.

Reference: BB 6.2.5; MS 5.15.2.10.c.

3.1.5 Command Language (cont.)

-7 A command language should provide flexibility, permitting the user to assign personal names to files, frequently used command sequences, etc.

Comment: Frequently used commands should be made easy to accomplish. Where users will differ in the frequency of the commands they use, the designer should provide for flexibility in command naming.

-8 A command language should be supported by whatever computer processing is necessary so that the user can manipulate data without concern for internal storage and retrieval mechanisms.

Example: The user should be able to request display of a file by name alone, without having to enter any further information such as file location in computer storage.

Comment: Where file names are not unique identifiers, the computer should be programmed to determine whatever further context is necessary for identification, automatically in relation to the current transaction sequence, or perhaps by asking the user to designate a "directory" defining a subset of files of current interest.

-9 The features of a command language should be designed in groups (or "layers") for ease in learning and use.

Comment: The fundamental core, or bottom layer, of the language should be the easiest, allowing use of the system by people with little training and/or limited needs. Successive layers of the command language can then increase in complexity for users with greater skills.

Reference: Reisner, 1977.

-10 The user should be able to request prompts as necessary to determine required parameters in a command entry, or to determine available options for an appropriate next command entry.

Reference: MS 5.15.1.4.5.

3.1.5 Command Language (cont.)

-11 When command entries are prompted automatically, it should be possible for an experienced user to key a series of commands at one time ("command stacking") so as to shortcut the prompting sequence.

Reference: BB 6.8.

See also: Section 3.2.

-12 Insofar as possible, the user should be able to enter commands without punctuation.

-13 If punctuation is needed, perhaps as a delimiter to distinguish optional parameters, or the separate entries in a stacked command, one standard symbol should be used consistently for that purpose, preferably the same symbol (slash) used to separate a series of data entries.

See also: 1.4-4, 3.2-18.

-14 Neither the user nor the computer program should have to distinguish between single and multiple blanks in a command entry.

Comment: People cannot be relied upon to pay careful attention to such details. The computer should handle them automatically, e.g., ensuring that two spaces follow every period in text entry, adding spaces needed to justify lines of text, etc.

-15 When command entries are subject to misinterpretation (as in the case of voice input), or when an interpreted command may have disruptive consequences, the user should be given an opportunity to review and confirm a displayed interpretation of the command before it is executed.

Comment: For beginning users, it might be desirable to permit review of interpreted commands for every transaction. Skilled users, however, should be able to suppress such routine review.

3.1.5 Command Language (cont.)

-16 When a command entry is not recognized, the computer should initiate a clarification dialogue, rather than rejecting the command outright.

Comment: Poorly stated commands should not simply be rejected. Instead, the computer should be programmed to guide the user toward a proper formulation, preserving the faulty command for reference and modification, and not require the user to re-key the entire command just to change one part.

3.1.6 Query Language

-1 Query language dialogue should be considered as a specialized sub-category of general command language for tasks emphasizing unpredictable information retrieval (as in many analysis and planning tasks), with moderately trained users and fast computer response.

Comment: All recommendations for command language design would apply equally to query languages, with the addition of some more specific guidelines listed below.

Reference: Ehrenreich, 1981.

-2 When the organization of the computer data base is reflected in the query language, that organization should match the data structure perceived by users to be natural.

Comment: The users' natural perception of data organization can be discovered through experimentation or by survey.

Reference: Durding, Becker and Gould, 1977.

-3 One single representation of the data organization should be established for use in query formulation, rather than multiple representations.

Comment: Beginning or infrequent users may be confused by different representational models.

3.1.6 Query Language (cont.)

-4 The need for quantificational terms in query formulation should be minimized.

Exception: "no" or "none".

Comment: People have difficulty in using quantifiers unambiguously. When quantifiers must be used, it may be desirable to have the user select the desired quantifier from a set of sample statements so worded as to maximize their distinctiveness.

-5 Use of operators subject to frequent semantic confusion, such as "or more" and "or less", should be minimized.

Example: A user should not have to convert "over 50 years old" into "51 or more".

3.1.7 Natural Language

-1 Unconstrained natural language dialogue should not be considered for USI design at this time; natural language may find future use in applications where task requirements are broad ranging and poorly defined, where little user training can be provided, and where computer response will be fast.

Comment: Computer processing of natural language is now being developed on an experimental basis. For current applications where task requirements are well defined, other types of dialogue will prove more efficient.

Reference: Shneiderman, 1981.

3.1.8 Graphic Interaction

-1 Graphic interaction should be considered as a supplement to other forms of man-machine dialogue where special task requirements exist; effective implementation of graphic capabilities will require very fast computer response.

3.2 Transaction Selection

-1 The sequence of transaction selections should generally be dictated by the user's choices and not by internal computer processing constraints.

Example: A data entry clerk should be able to enter items in whatever order they are available, when order is variable, as in taking reservation data by telephone.

Comment: In some cases this means that the computer may have to store current inputs until they become relevant to subsequent data processing.

Comment: When a logical sequence of transactions can be determined in advance, USI design might encourage and help a user to follow that sequence. Guidance may be desirable though constraint is not.

Reference: PR 4.6.7.

See also: 3.0-1, 3.0-5, 3.0-6.

-2 An initial menu of control options should always be available for user selection, to serve as a "home base" or consistent starting point for control inputs at the beginning of a transaction sequence.

Comment: Such a starting point is helpful even when all dialogue is user-initiated. This capability can be implemented as an OPTIONS function key, or as an explicit control option on every display, or as a generally available implicit option, or as a consistent default for a null control input.

Reference: BB 4.1; PR 3.3.16.

3.2 Transaction Selection (cont.)

-3 The general OPTIONS display should show primary control inputs grouped, labeled and ordered in terms of their logical function, frequency and criticality of use, following the guidelines provided for menu selection.

See also: Section 3.1.3.

-4 The user should be able to make at least some sequence control inputs directly at any step in a transaction sequence (i.e., from any display frame) without having to return to a general options display.

Comment: In particular, the user should not have to remember data or control codes given on one display for later input on a different display.

-5 Information should be provided the user concerning control options specifically appropriate at any step in a transaction sequence, either incorporated in the display or else available through a request for HELP.

Reference: MS 5.15.3.5.

-6 Control options that are generally available at any step in a transaction sequence should be treated as implicit options, i.e., need not be included in a display of step-specific options; frequently used implicit options should be input by function keys.

Comment: For applications involving experienced users, implicit options may be input by command entry.

See also: 3.1.4-2.

-7 When selection among displayed options is to be accomplished by pointing, the cursor should be placed automatically on the first (most likely) option at initial display generation.

-8 When selection among displayed options is to be accomplished by keyed entry of a corresponding code, the cursor should be placed automatically in the command entry area at initial display generation.

Reference: PR 4.7.1.

3.2 Transaction Selection (cont.)

-9 When displayed options can be selected by code entry, the code associated with each option should be included on the display in some consistent, identifiable manner.

Example: In many applications an equal sign can be used for this purpose, as N=NEXT PAGE, P=PREV PAGE, etc.

-10 The wording of step-specific control options should reflect the current concerns and likely questions of the user at that step in a transaction sequence.

Reference: MS 5.15.2.10.d.

-11 The user should not be offered control options that he cannot take.

See also: 3.1.3-17.

-12 If a default for a null control input is defined for any step in a transaction sequence, that default should be consistent in USI design, or else indicated in the display of step-specific control options.

Example: In menu selection, a null entry might always default to the first of the displayed options.

Reference: EG 4.2.4.

-13 When control input is accomplished by command entry, the user should have some consistent means to request prompting for input options or parameter values not already shown on the display.

Example: Keying a question mark in the command entry area would be a satisfactory method of requesting prompts, or else using an explicitly labeled HELP function key.

Comment: In some applications it may be desirable to let an inexperienced user simply choose a general "prompt mode" of operation, where any command entry produces automatic prompting of (required or optional) parameters and/or succeeding input options.

Reference: Demers, 1981.

3.2 Transaction Selection (cont.)

-14 At any step in a defined transaction sequence, if there are no alternative step-specific control options, then a consistent command should be used to continue to the next step.

Example: NEXT, STEP or CONTINUE might be suitable names for this function.

Exception: If data entry is involved, then an explicit ENTER command should be used.

Reference: PR 4.11.

-15 When control input involves command entry, the user should be permitted to key a sequence of codes for option selection as a single "stacked" command.

Example: In particular, the user should be able to enter stacked commands from the initial menu of general OPTIONS, so that an experienced user can make any specific control input the first step in a transaction sequence.

Example: Command stacking may be helpful when a user is being prompted to enter a series of parameter values, and knows in advance what several succeeding prompts will request and what values to enter.

Comment: Command stacking will permit a transition from simple step-by-step control input by novice users, as in menu selection and question-and-answer dialogues, to the entry of extended command-language statements by experienced users; command stacking is especially helpful in time-shared systems where computer response to any user input may be slow.

Reference: EG 6.2, 6.2.1; PR 2.6, 4.7.3; Palme, 1979.

See also: 3.1.5-11.

-16 In command stacking, user inputs should be in the same order as they would normally be made in a succession of separate command entry actions.

Reference: EG 6.2.1.

3.2 Transaction Selection (cont.)

-17 In command stacking, acceptable user inputs should include command names or their abbreviations or defined codes; if stacked command inputs are potentially ambiguous, the computer should display the interpreted command sequence for user confirmation or correction.

Reference: EG 6.2.1.

-18 In command stacking, if some special symbol must be used to separate command entries, then one standard symbol should be adopted for that purpose, preferably the same symbol (slash) used as a delimiter for sequential data entries.

Reference: EG 6.2.1.

See also: 1.4-4, 3.1.5-13.

-19 If a stacked command results in only partial completion of a menu selection sequence (i.e., if further user selections must be made), then the appropriate next menu display should be presented to guide completion of control input.

Reference: PR 4.7.3.

-20 Flexibility in transaction selection should be provided by permitting the user to assign a single name to a defined series of control inputs, and then use this new "macro" for subsequent command entry.

Comment: In this way the user can make frequently required but complicated tasks easier to accomplish, when the USI designer has failed to anticipate the particular need.

Reference: Demers, 1981.

3.3 Interrupt

-1 Flexibility in control should be provided by permitting the user to interrupt, defer or abort a current transaction sequence, in consistently defined ways appropriate to specific task requirements.

Comment: For an experienced user, it may be desirable to permit multi-tasking of the computer, so that a transaction involving slow data processing can be accomplished as "background" to interim shorter transactions. In such a case, of course, the computer must be programmed to signal completion of the background transaction.

Comment: Provision of flexible interrupt capabilities for the user will generally require some sort of suspense file or other buffering in software design. Such capabilities are valuable, however, in permitting the user to correct mistaken entries, and in permitting the computer to require user confirmation of potentially destructive entries.

Reference: PR 3.3.16, 3.3.17.

-2 If different degrees of interruption in sequence control are provided, they should be accomplished by differently named control options.

Comment: As a negative example, it would not be good design practice to provide a single ESCAPE key which has different effects depending upon whether it is pushed once or twice; the user may be confused by such expedients, and uncertain about what action has been taken and its consequences.

-3 If appropriate to sequence control, a CANCEL option should be provided, which will have the consistent effect of regenerating the current display without processing any interim changes made by the user.

Comment: In effect, interim entries would be erased.

See also: 1.4-2.

3.3 Interrupt (cont.)

-4 If appropriate to sequence control, a BACKUP option should be provided, which will have the consistent effect of returning to the display entered in the last previous transaction.

Comment: Such a BACKUP capability will generally prove feasible only in the software design of well-defined transaction sequences, but will prove helpful when it can be provided.

Comment: BACKUP might be designed to include cancellation of any interim entries made in a pending transaction. Alternatively, pending entries might be preserved without processing. USI design should be consistent in this regard.

Reference: MS 5.15.1.2.6.

See also: 1.4-2.

-5 If appropriate to sequence control, a RESTART option should be provided, which will have the consistent effect of returning to the first display in a defined transaction sequence, permitting the user to review a sequence of entries and make necessary changes.

Comment: As an extension of the BACKUP capability, RESTART is useful only in well-defined transaction sequences such as step-by-step data entry in a question-and-answer dialogue.

Comment: RESTART might be designed to include cancellation of any interim entries made in a pending transaction. Alternatively, pending entries might be preserved without processing. USI design should be consistent in this regard.

See also: 1.4-2.

-6 If appropriate to sequence control, an ABORT option should be provided, which will have the consistent effect of cancelling all entries in a defined transaction sequence; when data entries or changes will be nullified by an ABORT action, the user should be asked in an advisory message to CONFIRM the ABORT.

Comment: An ABORT action, combining the functions of RESTART and CANCEL, is again relevant only to well-defined transaction sequences, specifically those with a recognized beginning.

Reference: BB 1.8.

3.3 Interrupt (cont.)

-7 If appropriate to sequence control, an END option should be provided, which will have the consistent effect of concluding a repetitive transaction sequence and returning control to a general OPTIONS menu.

Example: In routine data entry, where the end of one transaction is designed to lead automatically to the beginning of the next transaction, the user needs some control input such as END to signal when a batch of transactions has been completed.

Comment: END can be implemented by whatever means are appropriate to the dialogue design, i.e., by menu selection, command entry, or function key.

Reference: EG 4.2.10.

See also: 3.0-15.

3.4 Context Definition

-1 Sequence control software should be designed to maintain context for the user throughout the series of transactions comprising a task, recapitulating previous inputs affecting present actions, and indicating currently available options where appropriate.

See also: 1.8-3, 3.0-17.

-2 In tasks where transaction sequences are variable, the user should be able to request a displayed list of prior entries if needed to determine present status.

Comment: Such a capability may not be needed for routine transactions if they are designed in such a way that each step identifies its predecessors explicitly, although even in those circumstances a user may be distracted and at least momentarily become confused.

Reference: EG 4.2.7.

3.4 Context Definition (cont.)

-3 Insofar as possible, sequence control software should be designed to carry forward a representation of the user's knowledge base and current activities; the user should not have to re-enter previously entered data relevant to current control inputs.

Example: If data have just been stored in a named file, then the user should be able to request a printout of that file without having to re-enter its name.

Exception: If transactions involving contextual interpretation would have destructive effects (e.g., data deletion), then the interpreted command should be displayed first for user confirmation.

Comment: The software logic supporting contextual interpretation of control inputs need not be perfect in order to be helpful. When ambiguity results, it may still be easier for the user occasionally to review and correct an interpreted command than always to generate a complete command initially.

Reference: MS 5.15.2.9; PR 2.3.

-4 When context for sequence control is established in terms of a defined operational mode, then some means should be provided to remind the user of the current mode and other pertinent information.

Example: If text is displayed in an editing mode, then a caption might indicate EDIT as well as the name of the displayed text; if an INSERT mode is selected for text editing, then some further displayed signal should be provided.

Reference: EG 4.2.1.

3.4 Context Definition (cont.)

-5 The value of any control parameter(s) currently operative should be displayed for user reference.

Comment: This practice is helpful even when the user selects all parameters himself, since he may well forget them, particularly if his task activities are interrupted.

Comment: When there are a large number of currently operative control parameters, it may prove impractical to display them continuously. In such a case, it may suffice to list them on an auxiliary HELP display accessed by user option.

Reference: MS 5.15.3.5.b.

-6 Whatever information is given the user to provide context for sequence control should be distinctive in location and format, and consistently displayed from one transaction to the next.

Reference: MS 5.15.3.6, 5.15.4.4.

3.5 Error Management

-1 The computer software should deal appropriately with all possible control inputs, correct and incorrect, without inducing errors in sequence control.

Example: If the user selects a function key that is invalid at a particular step in a transaction sequence, no action should result except display of an advisory message indicating what functions are appropriate at that point.

Comment: For certain routine and easily recognized errors, such as trying to tab beyond the end of a line, a simple auditory signal ("beep") may be sufficient computer response.

Reference: PR 4.12.4.5.

See also: 3.1.4-10.

3.5 Error Management (cont.)

-2 The user should be able to edit an extended command during its composition, by backspacing and rekeying, before taking an explicit action to ENTER the command.

Comment: Users can often recognize errors in keyed input prior to final entry.

Reference: EG 5.4.

See also: 1.4-2.

-3 If an element of a command entry is not recognized, or logically inappropriate, sequence control software should prompt the user to correct that element without having to re-enter the entire command.

Example: A faulty command can be retained in the command entry area of the display, with the cursor automatically positioned at the incorrect item, plus an advisory message describing the problem.

Reference: BB 1.3; EG 4.2.2, 4.2.3; MS 5.15.1.2.1, 5.15.1.2.7.b.

-4 If an error is detected in a stacked series of command entries, USI design should be consistent in how this is handled, from one transaction sequence to another.

Comment: It may help the user if the commands are executed to the point of error, or it may not. In most applications, partial execution will probably prove desirable. The point is that a considered USI design decision should be made and then followed consistently.

Reference: BB 1.5; EG 5.6; PR 4.7.3.

-5 If only a portion of a stacked command can be executed, that problem should be indicated to the user with appropriate guidance to permit completion of the intended control input.

See also: 3.2-19.

3.5 Error Management (cont.)

-6 The ENTER action for command entry should be the same as that for data entry; direct selection of menu options should also require some explicit ENTER action.

Comment: When a common action is used for both data entry and command entry, there will be less likelihood of user confusion and error.

See also: 1.0-8, 1.1-5, 3.0-5.

-7 When a user completes correction of an error, whether of a command entry or data entry, the user should be required to take an explicit action to re-enter corrected inputs; the new ENTER action should be the same as whatever action was appropriate to make that input originally.

Reference: PR 4.12.4.6.

-8 When a default value is included in command entry, it may be helpful to recapitulate the command in its fully interpreted form for user confirmation; if this practice is followed, it should be done consistently.

-9 When a control input will cause any extensive change in stored data, procedures and/or system operation, and particularly if that change cannot be easily reversed, the user should be notified and required to confirm the action before it is implemented.

Reference: BB 1.10; EG 4.1.2, 4.2.8; MS 5.15.1.2.3, 5.15.1.2.7.c.

-10 The prompt for CONFIRM action should be worded in such a way that potential data loss is clearly stated.

Example: CONFIRM DELETION OF ENTIRE AIRFIELD FILE may give adequate warning; CONFIRM DELETE ACTION does not.

3.5 Error Management (cont.)

-11 User confirmation of a control input or data entry should be accomplished with an explicitly labeled CONFIRM function key, different from the ENTER key.

Comment: Confirmation should not be accomplished by pushing some other key twice.

Comment: Some USI designers recommend that in special cases confirmation should be made more difficult still, e.g., by keying the separate letters in C-O-N-F-I-R-M. A better approach might be to make any user action (including over-hasty confirmation) immediately reversible, a feature now available in some current USI designs.

See also: 3.1.4-4, 3.1.4-14.

-12 When the user signals that he is ready to log off, sequence control software should check pending transactions and, if data loss seems probable, should display an advisory message requesting confirmation.

Example: CURRENT DATA ENTRIES HAVE NOT BEEN FILED; SAVE IF NEEDED, BEFORE CONFIRMING LOGOFF.

Comment: The user will sometimes suppose that a job is done before he has taken necessary final actions.

-13 When a data entry transaction has been completed and errors detected, sequence control logic should permit direct, immediate correction by the user.

Comment: It is helpful to correct data entry errors at the source, i.e., while the user still has the entry in mind and/or source documents at hand.

Comment: For transactions involving extended entry of multiple items, computer checking might be invoked by separate entry of each page (or section) of data.

Reference: PR 2.5.

See also: 1.7-4.

3.5 Error Management (cont.)

-14 The user should be able to return easily to previous steps in a transaction sequence in order to correct an error or make any other desired change.

Reference: MS 5.15.1.2.6.

See also: Section 3.3.

-15 When considerations of data security do not prohibit, the user should be able to change any data that are currently displayed.

Comment: The user should not have to specify that he wants to make changes in advance of calling for a display. That practice may be simpler for the software designer, but is confusing for the user.

3.6 Alarms

-1 In many applications, particularly those involving monitoring and process control, the user (or some authorized supervisor) should be permitted to define conditions, in terms of variables and values, that will result in automatic generation of alarm messages.

Example: The nurse in charge of an intensive care monitoring station might need to specify for each patient warning signals when blood pressure ("variable") exceeds or falls below defined levels ("values").

Exception: Situations where alarm conditions must be pre-defined by functional, procedural, or perhaps even legal requirements, such as violation of aircraft separation in air traffic control.

-2 Alarm signals and messages may take a variety of forms, but should be distinctive and consistent for each class of events.

Comment: The user might be permitted to define the nature of each alarm as well as its initiating event.

3.6 Alarms (cont.)

-3 The user should be provided a simple, consistent means of acknowledging and turning off non-critical alarm signals.

Example: A function key labeled ALARM ACK would suffice for that purpose.

Reference: MS 5.15.4.6.1.a.

-4 The user may be required to take more complicated actions in order to respond to critical alarms, and to acknowledge special alarms in special ways; but such special acknowledgment actions should be designed so that they will not inhibit or slow remedial user response to a critical initiating condition.

3.7 Design Change

(no guidelines presently available)

APPENDIX E

REFERENCES FOR DESIGN GUIDELINES

Anyone involved in compilation of USI design guidelines must begin and end by acknowledging the significant contributions of other people. This undertaking is truly a collaborative effort, as each new student of the field builds upon the work of predecessors and colleagues. It is good that this is so. All USI design guidelines are based in some degree on judgment, and often the joint judgment of multiple contributors will prove sounder than the views of just one person.

Most of the guidelines presented in this report were not invented here, but were built on contributions by other people. Where the idea for a guideline came from a particular source, or is supported by prior recommendations, an annotation of appropriate references has been included for that guideline. Such annotation offers credit, where credit is due. More importantly, cited references permit anyone questioning a particular guideline to explore its antecedents, perhaps to gain a better understanding of what is intended.

Citation of references does not necessarily mean that previous author(s) would agree with the wording of a guideline presented here. In some instances, the wording of a previously published guideline has been preserved. In many more cases, however, proposed guidelines have been re-worded here, and sometimes revised drastically.

Revision of published guidelines has not been undertaken capriciously, but only after careful consideration. During 1981, guidelines for data entry and for sequence control functions (Appendices B and D) were distributed in a previous report to the USI Guidelines Group. Replies were received from nine members of the group:

Sara R. Abbott	Union Carbide Corporation
Christopher J. Arbak	Systems Research Laboratories, Inc.
J. David Beattie	Ontario Hydro
Kent B. Davis	Litton Data Command Systems
Richard M. Kane	Essex Corporation
Lorraine F. Normore	Ohio State University Human Performance Center
Steven P. Rogers	Anacapa Sciences, Inc.
Eric M. Schaffer	Modern Human Resources, Inc.
John C. Thomas	IBM Corporation

These reviewers provided ratings for the guidelines, and offered suggestions and critical comments. Overall, 210 guidelines were rated, 79 for data entry and 131 for sequence control functions. Every guideline received mixed ratings, though some more mixed than others. Every guideline was judged useful by someone; there was no guideline that was judged useful by everyone. Here is a summary of the ratings:

guideline rated "useful"	59 %
guideline rated "fairly useful"	28
guideline rated "trivial"	2
guideline is wrong	2
guideline should be revised	6
guideline should be eliminated	1

In addition to the ratings, 209 specific comments or criticisms were offered, pertaining to 127 of the guidelines. As a result of those comments, changes were made to 123 of the guidelines, as published in the present report:

	Added	Re-worded	Deleted
guidelines	3	56	2
examples	6	3	
exceptions	4	2	
comments	41	5	
references	1		

To the extent that those revisions have improved the published guidelines, these reviewers deserve recognition. It may be true, however, that none of these reviewers would agree with all of the resulting guidelines, since opposed opinions could not always be effectively resolved.

Previously published guidelines were also reviewed by students in a course on Human Performance at the University of Michigan, under the tutelage of Paul Green, and several changes to guidelines resulted from suggestions by student reviewers. Further changes were recommended by Arlene F. Aucella at MITRE.

It is clear that critical review of USI design guidelines should continue. No guideline proposed here is worded so perfectly that it cannot be improved. And the general need persists for more illustrative examples, more completely noted exceptions, more explanatory comment, and more references. References are particularly needed to aid the review process, in cases where reviewers must try to resolve conflicting judgment.

In this report, a beginning has been made in citing source material for guidelines. In the guidelines compilation in previous appendices of this report, references to general sources for guidelines have been given in abbreviated form, with a two-letter code followed by the paragraph or item number in the referenced document. Four general sources are referenced in this way:

BB = Brown, Burkleo, Mangelsdorf, Olsen and Williams, 1981

EG = Engel and Granda, 1975

MS = MIL-STD-1472C, 1981

PR = Pew and Rollins, 1975

References to more specific source documents have been cited in conventional form by author and date. Those specific document references are listed in the pages that follow. This is obviously just a small sample of potentially relevant sources, including primarily recent references immediately familiar to the author. A concerted effort will be made to expand this listing in order to provide more comprehensive coverage in the future.

Perhaps you can help in this work by proposing new guidelines, or by suggesting improvements to those published here, or by citing examples, exceptions, and references. If so, please copy the change form included at the back of this report, and use it for your contribution.

REFERENCES FOR GUIDELINES

Aretz, A. J. and Kopala, C. J. Automatic return in multifunction control logic. In Proceedings of the 25th Annual Meeting. Santa Monica, California: Human Factors Society, 1981, 254-256.

Brown, C. M., Burkleo, H. V., Mangelsdorf, J. E., Olsen, R. A., and Williams, A. R., Jr. Human Factors Engineering Standards for Information Processing Systems. Sunnyvale, California: Lockheed Missiles and Space Company, Inc., 15 June 1981.

Bury, K. F., Boyle, J. M., Evey, R. J., and Neal, A. S. Data manipulation on a visual display terminal: Windowing or scrolling? Human Factors, in press.

Campbell, A. J., Marchetti, F. M., and Mewhort, D. J. K. Reading speed and text production: A note on right-justification techniques. Ergonomics, 1981, 24(8), 633-640.

Demers, R. A. System design for usability. Communications of the ACM, 1981, 24(8), 494-501.

Dray, S. M., Ogden, W. G., and Vestewig, R. E. Measuring performance with a menu-selection human-computer interface. In Proceedings of the 25th Annual Meeting. Santa Monica, California: Human Factors Society, 1981, 746-748.

Durding, B. M., Becker, C. A., and Gould, J. D. Data organization. Human Factors, 1977, 19(1), 1-14.

Ehrenreich, S. L. Query languages: Design recommendations derived from the human factors literature. Human Factors, 1981, 23(6), 709-725.

Engel, S. E. and Granda, R. E. Guidelines for Man/Display Interfaces, Technical Report TR 00.2720. Poughkeepsie, New York: IBM, December 1975.

Gade, P. A., Fields, A. F., Maisano, R. E., and Marshall, C. F. Training approaches and data entry methods for semi-automated command and control systems. In Proceedings of the 24th Annual Meeting. Santa Monica, California: Human Factors Society, 1980, 416-420.

Gregory, M. and Poulton, E. C. Even versus uneven right-hand margins and the rate of comprehension in reading. Ergonomics, 1970, 13, 427-434.

Hamill, B. W. Experimental document design: Guidebook organization and index formats. In Proceedings of the 24th Annual Meeting. Santa Monica, California: Human Factors Society, 1980, 480-483.

Hanson, R. H., Payne, D. G., Shiveley, R. J., and Kantowitz, B. H. Process control simulation research in monitoring analog and digital displays. In Proceedings of the 25th Annual Meeting. Santa Monica, California: Human Factors Society, 1981, 154-158.

Hollingsworth, S. R. and Dray, S. M. Implications of post-stimulus cueing of response options for the design of function keyboards. In Proceedings of the 25th Annual Meeting. Santa Monica, California: Human Factors Society, 1981, 263-265.

Martin, J. Design of Man-Computer Dialogues. Englewood Cliffs, New Jersey: Prentice-Hall, 1973.

Miller, D. P. The depth/breadth tradeoff in hierarchical computer menus. In Proceedings of the 25th Annual Meeting. Santa Monica, California: Human Factors Society, 1981, 296-300.

Miller, R. B. Response time in user-system conversational transactions. In Proceedings of the AFIPS Fall Joint Computer Conference, 1968, 267-277.

Morrill, C. S. and Davies, B. L. Target tracking and acquisition in three dimensions using a two-dimensional display surface. Journal of Applied Psychology, 1961, 45, 214-221.

Moses, F. L. and Ehrenreich, S. L. Abbreviations for automated systems. In Proceedings of the 25th Annual Meeting. Santa Monica, California: Human Factors Society, 1981, 132-135.

MIL-STD-1472C. Military Standard: Human Engineering Design Criteria for Military Systems, Equipment and Facilities. Washington: Department of Defense, 2 May 1981.

Noyes, L. The positioning of type on maps: The effect of surrounding material on word recognition. Human Factors, 1980, 22(3), 353-360.

Palme, J. A human-computer interface for non-computer specialists. Software -- Practice and Experience, 1979, 9, 741-747.

Pew, R. W. and Rollins, A. M. Dialog Specification Procedures, Report 3129 (revised). Cambridge, Massachusetts: Bolt Beranek and Newman, 1975.

Ramsey, H. R. and Atwood, M. E. Human Factors in Computer Systems: A Review of the Literature, Technical Report SAI-79-111-DEN. Englewood, Colorado: Science Applications, Inc., September 1979. (NTIS No. AD A075 679)

Reisner, P. Use of psychological experimentation as an aid to development of a query language. IEEE Transactions on Software Engineering, 1977, SE-3, 218-229.

Seibel, R. Data entry devices and procedures. In Van Cott, H. P. and Kinkade, R. G. (Eds.) Human Engineering Guide to Equipment Design. Washington: U. S. Government Printing Office, 1972, 311-344.

Shneiderman, B. A note on human factors issues of natural language interaction with database systems. Information Systems, 1981, 6(2), 125-129.

Smith, S. L. Angular estimation. Journal of Applied Psychology, 1962, 46, 240-246. (a)

Smith, S. L. Color coding and visual search. Journal of Experimental Psychology, 1962, 64(5), 434-440. (b)

Smith, S. L. Color coding and visual separability in information displays. Journal of Applied Psychology, 1963, 47(6), 358-364. (a)

Smith, S. L. Man-computer information transfer. In Howard, J. H. (Ed.) Electronic Information Display Systems, 284-299. Washington: Spartan Books, 1963. (b)

Smith, S. L., Farquhar, B. B., and Thomas, D. W. Color coding in formatted displays. Journal of Applied Psychology, 1965, 49(6), 393-398.

Smith, S. L. and Goodwin, N. C. Computer-generated speech and man-computer interaction. Human Factors, 1970, 12(2), 215-223.

Smith, S. L. and Goodwin, N. C. Alphabetic data entry via the Touch-Tone pad: A comment. Human Factors, 1971, 13(2), 189-190. (a)

Smith, S. L. and Goodwin, N. C. Blink coding for information display. Human Factors, 1971, 13(3), 283-290. (b)

Smith, S. L. and Goodwin, N. C. Another look at blinking displays. Human Factors, 1972, 14(4), 345-347.

Smith, S. L. and Thomas, D. W. Color versus shape coding in information displays. Journal of Applied Psychology, 1964, 48(3), 137-146.

Tullis, T. S. An evaluation of alphanumeric, graphic, and color information displays. Human Factors, 1981, 23(5), 541-550.

Whalley, P. C. and Fleming, R. W. An experiment with a simple recorder of reading behaviour. Programmed Learning and Educational Technology, 1975, 12(2), 120-124.

Wright, P. and Reid, F. Written information: Some alternatives to prose for expressing the outcomes of complex contingencies. Journal of Applied Psychology, 1973, 57(2), 160-166.

USI Design Guidelines -- Changes/Additions

Code number, if referring to a guideline in this report: _____

Proposed wording for guideline: _____

_____

_____

_____

_____

Example: _____

_____

_____

_____

Exception: _____

_____

_____

_____

Comment _____

_____

_____

_____

References to previously published recommendations or supporting data:

_____

_____

_____

Please mail to:
Sidney L. Smith
The MITRE Corporation
Bedford, MA 01730 USA

Your name: _____

Date: _____


DATE
FILME
7 — 8